

DTIC FILE COPY

2

# NAVAL POSTGRADUATE SCHOOL

Monterey, California

AD-A209 437



DTIC  
ELECTE  
JUN 29 1989  
S D & D

## THESIS

IMPLEMENTATION OF A  
DISTRIBUTED OBJECT-ORIENTED  
DATABASE MANAGEMENT SYSTEM

Lynn A. Wyrick

March 1989

Thesis Advisor:

Valdis Berzins

Approved for public release; distribution is unlimited.

89 6 28 039

# REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b. OFFICE SYMBOL (If applicable) Code 52	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000		7b. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	
8a. NAME OF FUNDING, SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code)		10. SOURCE OF FUNDING NUMBERS	
		PROGRAM ELEMENT NO	PROJECT NO
		TASK NO	WORK UNIT ACCESSION NO
11. TITLE (Include Security Classification) IMPLEMENTATION OF A DISTRIBUTED OBJECT-ORIENTED DATABASE MANAGEMENT SYSTEM			
12. PERSONAL AUTHOR(S) Wyrick, Lynn A.			
13a. TYPE OF REPORT Master's Thesis	13b. TIME COVERED FROM TO	14. DATE OF REPORT (Year, Month, Day) March 1989	15. PAGE COUNT
16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	Distributed DBMS, Object-Oriented DBMS, concurrency, Database Implementation, Database Design, Distributed Architecture, NMPCDS, KBAS, (Knowledge Base Software Assistant)	
19. ABSTRACT (Continue on reverse if necessary and identify by block number) Distributed database management systems provide for more flexible and efficient processing. Research in object-oriented database management systems is revealing an abundance of additional benefits that cannot be provided by more traditional database management systems. The Naval Military Personnel Command (NMPC) is used as a case study to evaluate the requirements of transitioning from a centralized to a distributed database management system. Features and characteristics of both distributed and object-oriented database management systems are used to determine the appropriate configuration for different application environments. The distributed and object-oriented concepts are evaluated in detail in order to allow an organization to appropriately select the type of system to meet their needs. Transition requirements for NMPC, in particular, are identified and a suggested plan of action is presented. <i>Keywords: Thesis</i>			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Prof. Valdis Berzins		22b. TELEPHONE (Include Area Code) (408)-646-2461	22c. OFFICE SYMBOL 52Be

Approved for public release; distribution is unlimited.

IMPLEMENTATION OF A DISTRIBUTED  
OBJECT-ORIENTED DATABASE MANAGEMENT SYSTEM

by

Lynn A. Wyrick  
Lieutenant, United States Navy  
B.S., Marquette University, 1984

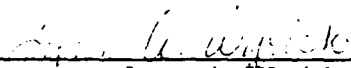
Submitted in partial fulfillment of the  
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

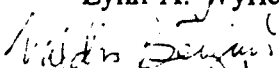
from the

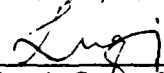
NAVAL POSTGRADUATE SCHOOL  
March 1989

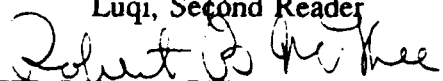
Author:

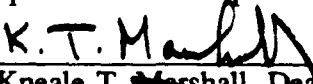
  
Lynn A. Wyrick

Approved by:

  
Valdis Berzins, Thesis Advisor

  
Luqi, Second Reader

  
Robert B. McGhee, Chairman  
Department of Computer Science

  
Kneale T. Marshall, Dean of  
Information and Policy Sciences

## ABSTRACT

Distributed database management systems provide for more flexible and efficient processing. Research in object-oriented database management systems is revealing an abundance of additional benefits that cannot be provided by more traditional database management systems. The Naval Military Personnel Command (NMPC) is used as a case study to evaluate the requirements of transitioning from a centralized to a distributed database management system. Features and characteristics of both distributed and object-oriented database management systems are used to determine the appropriate configuration for different application environments. The distributed and object-oriented concepts are evaluated in detail in order to allow an organization to appropriately select the type of system to meet their needs. Transition requirements for NMPC, in particular, are identified and a suggested plan of action is presented.



Accession For	
NTIS CRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

## TABLE OF CONTENTS

<b>I. INTRODUCTION</b> .....	1
<b>A. BACKGROUND</b> .....	1
<b>B. OBJECTIVES</b> .....	3
<b>C. RESEARCH QUESTIONS</b> .....	4
1. Distributed Research .....	4
2. Object-oriented Research .....	4
<b>D. SCOPE, LIMITATIONS, ASSUMPTIONS</b> .....	5
1. Scope .....	5
2. Limitations .....	5
3. Assumptions .....	5
<b>E. LITERATURE REVIEW AND METHODOLOGY</b> .....	6
<b>F. SUMMARY OF FINDINGS</b> .....	6
<b>G. ORGANIZATION OF STUDY</b> .....	7
<b>II. BACKGROUND</b> .....	9
<b>A. OVERVIEW</b> .....	9
<b>B. ORGANIZATIONAL STRUCTURE</b> .....	10
<b>C. NMPC-4 PERSONNEL DISTRIBUTION FUNCTIONS</b> ...	11
1. Inventory Prediction and Allocation .....	12
2. Manning .....	12

3. Individual Assignments .....	12
D. CURRENT AUTOMATED INFORMATION SYSTEMS ..	14
E. CURRENT HARDWARE ENVIRONMENT .....	16
F. NMPDS TRANSITION PLAN .....	17
G. JUSTIFICATION FOR NMPDS TO TRANSITION TO A DBMS .....	18
H. CONCLUSION .....	19
III. DISTRIBUTED METHODS .....	21
A. OVERVIEW .....	21
1. Centralized Versus Distributed Systems .....	21
2. Distributed Definitions .....	23
B. EVALUATION OF DISTRIBUTED METHODS .....	25
1. Distributed Architectures .....	26
a. <i>Network-Oriented Operating System (NOS)</i> ..	27
b. <i>Network Access Processor (NAP)</i> .....	27
c. <i>In-House Configuration</i> .....	29
d. <i>Distributed System Environment</i> .....	30
e. <i>SHARD</i> .....	31
f. <i>Distributed Architecture Layers</i> .....	35
2. Process Handling Algorithms .....	35
a. <i>Specification of Time Dependencies</i> .....	36
b. <i>Distributed Scheduling</i> .....	38

<i>c. Task Response Time Model</i> . . . . .	40
<i>d. Evaluation of Process Handling Algorithms</i> . . . . .	44
3. Data Handling Algorithms . . . . .	45
C. CONCLUSIONS . . . . .	50
1. Management Issues . . . . .	50
<i>a. Performance Tuning</i> . . . . .	50
<i>b. Integration</i> . . . . .	51
2. NMPC Requirements . . . . .	52
IV. OBJECT-ORIENTED DBMSs . . . . .	54
A. INTRODUCTION . . . . .	54
B. FEATURES OF OBJECT-ORIENTED DBMSs . . . . .	57
C. EVALUATION OF OBJECT-ORIENTED SYSTEMS . . . . .	63
1. PROBE: An Object-Oriented, Extensible Database System . . . . .	63
2. Robust Generation of Unique Identifiers . . . . .	67
D. CONCLUSION . . . . .	69
V. KNOWLEDGE BASED SOFTWARE ASSISTANT . . . . .	72
A. INTRODUCTION . . . . .	72
B. KBSA DESIGN . . . . .	73
C. COMPARISON OF PROBE AND KBSA . . . . .	77
D. CONCLUSION . . . . .	78

<b>VI. TRANSITION AND <del>ISSUES</del> INTEGRATION</b>	79
<b>A. INTRODUCTION</b>	79
<b>B. HARDWARE</b>	80
1. Background	80
2. Requirements Contracts	83
3. Procedures	85
4. Desktop III	87
5. Alternative Considerations	89
6. Conclusions	90
<b>C. SOFTWARE</b>	92
<b>D. TELECOMMUNICATIONS</b>	97
<b>E. CONCLUSION</b>	99
<b>VII. CONCLUSION</b>	100
<b>A. SUMMARY</b>	100
<b>B. RECOMMENDATIONS</b>	101
<b>LIST OF REFERENCES</b>	106
<b>BIBLIOGRAPHY</b>	109
<b>INITIAL DISTRIBUTION LIST</b>	110



## LIST OF FIGURES

Figure 1	NMPC-47 ISM Support . . . . .	11
Figure 2	The Naval Military Personnel Distribution System . . . . .	16
Figure 3	NMPDS Hardware Configuration . . . . .	17
Figure 4	Levels of Architecture . . . . .	30
Figure 5	Access Control . . . . .	75
Figure 6	NMPDS Conversion Plan of Action . . . . .	104

# **I. INTRODUCTION**

## **A. BACKGROUND**

Technology available today provides support to organizations which are required to process large amounts of data. Database management systems were designed specifically to assist those organizations in efficiently managing their information. Until recently, database management systems have primarily been implemented in centralized environments, using a central data processing center, where users share system resources and data. New concepts in database management systems are being developed to improve on the technology currently in use. In particular distributed and object-oriented database management systems are models which are producing the most efficient and flexible results for moving toward the future. Organizations are faced with the problem of evaluating the database models and the software produced, and to determine the most appropriate choice in terms of their requirements in the years to come.

This thesis investigates the requirements of transitioning from a centralized database management system (DBMS) to a distributed, object-oriented DBMS at the Naval Military Personnel Command (NMPC). Distributed processing and distributed data allocation may be the most appropriate means to process information in the Navy's systems.

Object-oriented DBMSs are also being researched as an efficient, flexible and extensible way to implement database systems.

Distributed processing brings about special problems not found in centralized processing. The primary issues, data synchronization and integrity, will be addressed. There are a variety of distributed algorithms designed to maintain data integrity in a system where there may be site failures, communication failures, or timing delays. Some algorithms are much more complicated than others. They will be evaluated in terms of complexity and requirements for pure data integrity (i.e., is 100% data synchronization required?). NMPC's needs will be considered in terms of the complexity and efficiency provided by the algorithms.

A new concept in the area of database management is object-oriented database management systems. Object orientation has been identified as the key to extensibility for database systems. The object classes created allow a database to reach beyond traditional database applications. The data base is then defined in terms of the object classes created. This idea allows the users to define their own concepts in the database. This thesis investigates several systems which have been developed using object orientation. These systems are evaluated in terms of the requirements of the Naval Military Personnel Command. The level of effort required and benefits of the system are discussed.

The Naval Military Personnel Command is responsible for distributing the 600,000 naval personnel. This can be a rather complex job with many

factors influencing the decisions made. Automated support is currently in production in the form of applications programs written in a COBOL generator code.

The Knowledge Based Software Assistant (KBSA) developed by Honeywell Systems and Research Center is a distributed, object-oriented database management system. The KBSA is a tool developed to aid a team of programmers in developing large software projects. The requirements which drove development of the KBSA will be evaluated, the system will be studied in depth, and recommendations made concerning the application of a KBSA-like system for NMPC.

Transition requirements for NMPC to move from the current system to the proposed system will be addressed. Interface issues will also be discussed in terms of what an organization must consider in the process of designing a new system.

## **B. OBJECTIVES**

The Naval Military Personnel Command (NMPC-47) has developed four major automated information systems (AISs) in COBOL using the Sage APS Code Generator. These systems are used by the Distribution Department (NMPC-4). This thesis investigates the requirements of transitioning these projects from a centralized DBMS environment to a distributed DBMS at workstations, using NMPC as a case study, and makes recommendations about the direction NMPC may go. Object-oriented and

relational DBMSs will be compared, distributed systems will be evaluated, and transition requirements will be stated. This thesis attempts to address the issues involved that must be considered.

### **C. RESEARCH QUESTIONS**

This thesis investigates the definitions of concepts outlined in the following lists:

#### **1. Distributed Research**

- a. What is a distributed DBMS?
- b. What are its properties?
- c. Is a true distributed system practical?
- d. How much effort would be involved in the transition?
- e. What area would most of the effort be concentrated?
- f. What software should be used in terms of what is available to NMPC?

#### **2. Object-oriented Research**

- a. What is an object-oriented DBMS?
- b. How is it used?
- c. For what applications is it best suited?
- d. How can we combine the concepts of distributed and object-oriented databases?
- e. Is combining these concepts the best way to go for the future?

- f. How much effort is involved in the transition?
- g. Are the efforts worth the benefits?

## **D. SCOPE, LIMITATIONS, ASSUMPTIONS**

### **1. Scope**

The Model 204 (M204) database management system is a relational-like DBMS developed by the Computer Corporation of America (CCA). It has been partially implemented by the Distribution Support Division in a centralized environment. However, functions of the individual assignment and placement officers lend themselves to a distributed environment because they each work with their own sections of the databases. Research in object-oriented DBMSs is proving this area to be more beneficial than traditional DBMSs, more cost effective, and more adaptable to change. Object-oriented DBMSs will be evaluated and compared with Model 204. The results of the study will be combined into a policy management guide for NMPC.

### **2. Limitations**

There may be two limitations which affect the recommendations made in this thesis. The first is financial considerations. Budget cuts may prevent implementation of the DBMS. The second limitation is personnel. Loss of experienced personnel and reductions of billets will hinder the progression of the implementation by requiring new personnel to be trained and brought up to speed on the projects.

### **3. Assumptions**

The five major assumptions made in this thesis are: 1) NMPDS will transition to a DBMS, 2) The DBMS software recommended will be available on the market, 3) the hardware required to implement a distributed system will be available, 4) optical disk storage may be used so high resolution screens will be required to display the images, and 5) the Base Information Transfer System (BITS) will be implemented. (BITS is sponsored by OP-162 to carry out the transition of phone lines on Navy bases from analog to digital lines.)

## **E. LITERATURE REVIEW AND METHODOLOGY**

A major portion of the research effort was through review and comparison of sources of literature. An evaluation of the literature has led to the conclusions made in the thesis. Data was collected through a variety of technical manuals, academic papers, and system specifications in order to obtain a better understanding of distributed algorithms, distributed DBMSs, and object-oriented systems. Interviews were conducted with software developers, program managers, and database users to collect information not otherwise formally documented.

## **F. SUMMARY OF FINDINGS**

The object-oriented model of the database management system is indeed more flexible, extensible, and able to help solve more difficult applications

than traditional models. The research in this area has given every indication that this will become a more efficient and useful way to manage database systems. However, because the area is still under research, unforeseen problems may not have been discovered as yet. The systems thus far developed have been relatively application specific. Object-oriented DBMSs lend themselves nicely to non-traditional, more complicated applications. Relational database models were designed primarily to handle record processing. The relational model, and languages which support it, have been on the market for a few years and have been sufficiently tested. The applications at NMPC are primarily record processing applications, and therefore do not require the more sophisticated object-oriented model. The recommendation made in the thesis is to use a relational type of model for the DBMS to be used in a distributed environment. The organization is already familiar with and satisfied with Model 204, and can therefore build on knowledge already gained through the use of the system.

## **G. ORGANIZATION OF STUDY**

Chapter 2 will present the background of the problem. NMPC is responsible for the allocation and manning of Navy commands, and the distribution of naval personnel in support of the commands' needs. NMPDS is the automated support developed traditionally with COBOL and flat files. Justification for moving to a DBMS is provided in this chapter.



Chapter 3 investigates a number of methods for implementing distributed algorithms and their applications to data base management. It compares, contrasts, and analyzes these various methods in terms of actual software implementation. Guidelines are outlined for determining those applications suited for distributed processing, and some possible extensions made possible by this DBMS model.

Object-oriented DBMSs are reviewed in Chapter 4. Ongoing efforts in this field of study are analyzed. Methods and levels of effort required to implement an object-oriented database system are discussed. Object-oriented DBMS applications are compared to traditional DBMS applications and future extensions are discussed.

The Knowledge Based Software Assistant (KBSA) is used as a case study in Chapter 5 which combines the theories of distributed and object-oriented data base management systems. Level of effort required to implement the KBSA and benefits realized are reviewed.

Transition requirements in terms of hardware, software, and communications are presented in Chapter 6. This chapter includes acquisition of new ADP equipment, identifies any interface issues which will pertain to the NMPDS implementation of the DBMS.

The final chapter presents conclusions, recommendations, and constructs a rough plan of action for the NMPDS implementation of the database system.

## **II. BACKGROUND**

### **A. OVERVIEW**

Organizations must consistently look at new technology to find better, faster, and more efficient ways to process information within an organization. Database management systems were specifically designed to help manage data. There is a current trend toward distributed systems, which provide additional benefits by virtue of their distributed nature. The system currently in use, the Naval Military Personnel Distribution System (NMPDS) is using data files to store the data. A DBMS, Model 204, has been implemented for use with Ad Hoc queries and report generation. If NMPC integrates NMPDS fully to the Model 204 DBMS, then the organization will be operating in a centralized DBMS environment.

The Distribution Support Division (NMPC-47) is responsible for supporting the Distribution Department (NMPC-4) in the execution of personnel distribution. The Distribution Department, which consists of approximately one thousand members, is responsible for distributing over 600,000 Navy personnel. Prior to 1984, this complete process was handled by passing volumes of paper from desk to desk. The Distribution Support Division developed four automated information systems (AISs) to support the Distribution Department. Along with the development of these systems,

a proposal for a hardware and software procurement was written. The Navy awarded a contract for a series of IBM 4381 minicomputers, and the DBMS included in the award was CCA's Model 204 DBMS. NMPC-47 would like to develop a "strawman" to guide their planning issues for the next three to five years. This chapter will provide detail on the functions performed within the Distribution Department, the four automated information systems currently in existence, and a current reorganization planned by NMPC-47.

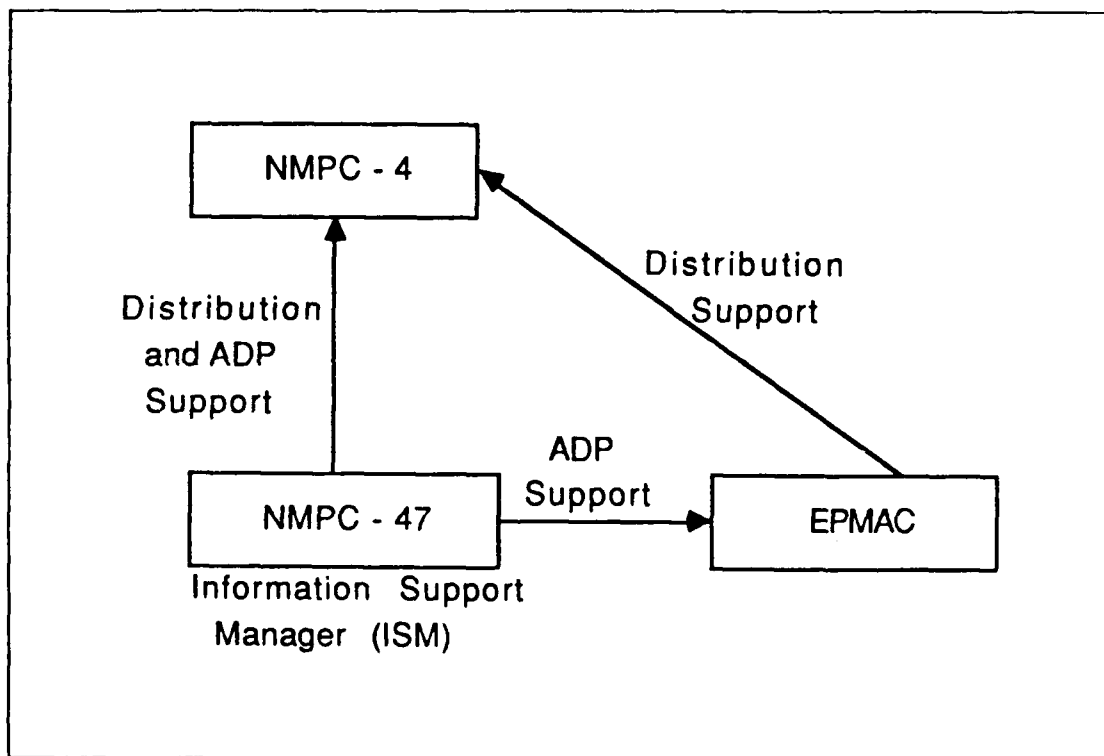
## **B. ORGANIZATIONAL STRUCTURE**

The Naval Military Personnel Command Distribution Department (NMPC-4) is responsible for assigning officer and enlisted active duty personnel in accordance with the needs of both the Navy and the individual's career; for controlling the manning of activities; and performing any additional functions as may be required.

The Distribution Support Division (NMPC-47) supports NMPC-4 by providing the ADP resources required for personal distribution management.

The Enlisted Personnel Management Center (EPMAC) is to provide centralized management support for active duty enlisted personnel, and to collect and disseminate manpower and personnel information to Navy activities. EPMAC reports to and works closely with NMPC to provide the Manning Control Authorities (MCA's) in the fleet with accurate manpower and personnel information. EPMAC provides their own ADP support and shares ADP resources with the Naval Reserve Personnel Command (NRPC).

NMPC and EPMAC work together to ensure the fleet is manned according to the number of available personnel, and that the readiness of the fleet is at an acceptable level. NMPC-47 is the Information Support Manager (ISM) for both NMPC-4 and EPMAC, and is therefore responsible for the support and coordination of ADP at both commands. Figure 1 illustrates the relationship between NMPC-4, NMPC-47 and EPMAC in terms of both distribution support and ADP support.



**Figure 1** NMPC-47 ISM Support

### **C. NMPC-4 PERSONNEL DISTRIBUTION FUNCTIONS**

The Distributions functions can be categorized into three areas: 1) Inventory Prediction and Allocation, 2) Manning, and 3) Individual

Assignments [Ref. 1]. These areas are described in the following paragraphs.

### **1. Inventory Prediction and Allocation**

The number of qualified people available to fill the number of billets (jobs) available determines the level of manning at Navy commands. People are constantly entering and leaving the Navy, so predicting the level of manning is required. This function ensures that each unit receives its share of qualified people, and is performed on a routine basis.

### **2. Manning**

Once a unit's "fair share" of personnel is determined, then the unit vacancy must be evaluated. The number of people onboard are compared to the number of billets at that command. Shortages and vacancies are categorized by "fill" time and priority indicators. These vacancies become a demand for personnel to be assigned to them.

### **3. Individual Assignments**

Individuals available for assignment are matched to those vacancies which best meet the individual's needs and the readiness demand of the unit. This information is fed back into the inventory process to keep that data base accurate.

Specifically, individuals are assigned through the following process. Personnel due to rotate a certain number of months ahead are earmarked as "available" for assignment. Jobs which will become vacant in the near future are "posted". This is the notification process to the detailers. These

vacant jobs have certain skills associated with them. A detailer chooses a person with those skills, or determines the training required to give an individual that skill. He must then ensure the training is available during the timeframe required; that is, between the rotation date and check in date, and makes reservations to the training. The "proposal" of the person to fill the job is then passed to the placement officers for approval. The proposed set of orders may be required to have approvals from other offices within the Distribution Department [Ref. 1].

Once the set of orders has been approved, the cost to move the member and his family is then estimated. Variables affecting the cost estimate are: distance moved (mileage), number of dependents, and enroute training. These cost estimates are used to compute annual budgets for PCS moves. This funding plays a major role in determining the amount of moves that may be executed by a detailer throughout a fiscal year.

The final step in the assignment process is to produce the set of orders for the individual. This process brings together all pertinent information about the transfer of duty into a single document. Generally, the orders contain detaching activity information, intermediate (training) information, ultimate duty station information, and accounting data. The orders may also contain any number of standard or special instructions.

#### **D. CURRENT AUTOMATED INFORMATION SYSTEMS**

The Distribution Support Division was tasked with supporting the functions of the Distribution Department through automated support. NMPC-47 designed and developed four systems: OAIS, EAIS, SPIRIT, and DMSS, which collectively are referred to as the Naval Military Personnel Distribution System (NMPDS), and are described in more detail in the following paragraphs. The systems in NMPDS support the management of training, allocation, and assignments of active duty officer and enlisted personnel. A system of rapid prototyping was used to develop these systems.

The Officer Assignment Information System (OAIS) was designed to support the distribution and assignment of officer personnel. This system was the first to go into production with its Surface Warfare Module in 1984. OAIS supports a "chop chain" which allows the detailers to route a set of orders from desk to desk electronically for approvals. It is now fully in production supporting all communities of officers and automatically producing the orders.

The Support Programs for Incentives, Retention and Training (SPIRIT) assignments was the next project to go into production. This system started out supporting training assignments (A-schools) for Recruits transferring out of boot camp (Recruit Training Centers). This training determines the rating or major skill enlisted personnel follow throughout their careers. The assignment process occurs weekly, and uses a linear program to optimize

the best recruit-skill match in a batch processing environment. SPIRIT also supports on-line training assignment reservations made to other Navy schools. These schools are attended enroute to new duty stations, and supplement knowledge already gained in a primary skill area. Both officer and enlisted detailers use SPIRIT to reserve training seats. The final area supported by SPIRIT is the Incentive and Retention Programs. Information is collected and processed through SPIRIT to manage the programs set up to retain sailors in the Navy with above average performance.

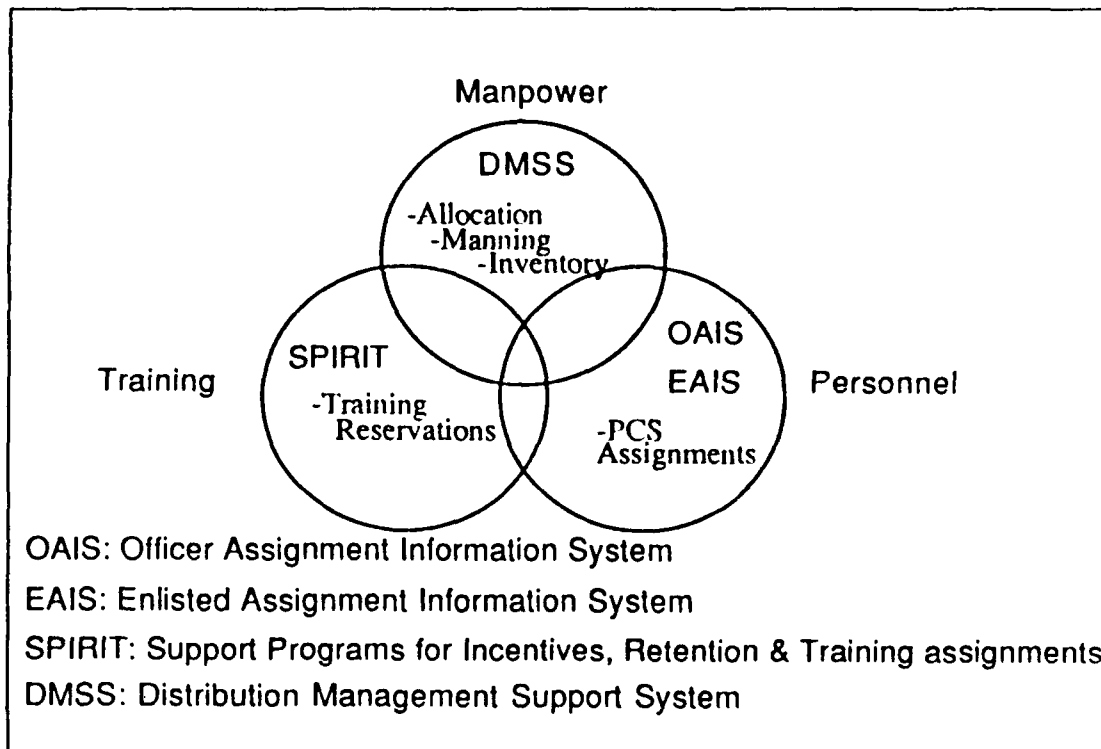
The Enlisted Assignment Information System (EAIS) supports the detailers in the distribution and assignment of enlisted personnel. Its Personnel Information Module (PIM) went into production in 1984, and the Assignment Decision Module and Order Production Module are scheduled to be in full production by 1990. This system has on-line communications with EPMAC, New Orleans, where the enlisted placement officers review assignments.

The Distribution Management Support System (DMSS) supports inventory prediction, allocation, and manning, which guide policy decisions in the individual assignments process. This system includes decision support system (DSS) modules to aid in the management of the following areas: Manning Plans, Projections, Personnel Requisitions, and Allocation & Nomination systems for both officer and enlisted personnel. The programs designed in this system deal with large amounts of data since the processing applies to Navy wide applications. These modules are run in a batch



environment and use terminals to display results of the processing on-line.

Figure 2 shows the relationship between the projects in NMPDS.



**Figure 2** The Naval Military Personnel Distribution System

## **E. CURRENT HARDWARE ENVIRONMENT**

The NMPDS is running on a series of IBM's located in Washington, DC, New Orleans, Louisiana, and Memphis, Tennessee. The current hardware configuration is depicted in Figure 3 and consists of five IBM 4381s in three different cities. The configuration is as follows:

Washington DC : Three 4381's

- 1) OAIS, EAIS, SPIRIT, 2) DMSS, Information Center
- 3) Maintenance

New Orleans

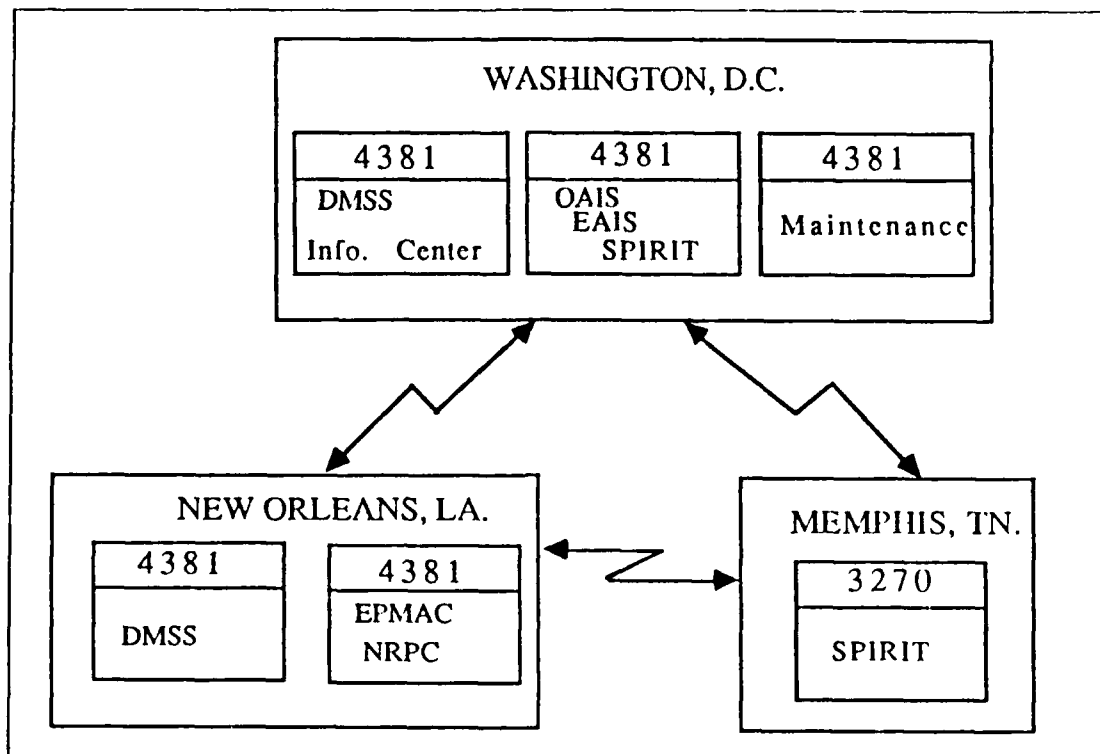
Two 4381's

- 1) DMSS, 2) EPMAC, NRPC

Memphis

3720 Communications Controller

- 1) CNTT, SPIRIT



**Figure 3** NMPDS Hardware Configuration

## F. NMPDS TRANSITION PLAN

The Distribution Support Division plans to reorganize NMPDS in order to provide for a more logical scheme for identifying, categorizing, and budgeting the systems [Ref. 1]. They will be organized more closely by function. OAIS, EAIS, SPIRIT and subsystems supporting the assignment function will be included in the Assignment Management Support System (AMSS). Systems supporting the allocation function will be aggregated under the Inventory Management Support Systems (IMSS), and those systems supporting the manning function will be grouped under the Manning Readiness Support System (MRSS). Finally, subsystems in the

NMPDS which do not directly support any of the distribution functions but do support other Manpower, Personnel and Training (MPT) processes will be addressed under the Manpower, Personnel and Training Support System (MPTSS). NMPC-47 intends to transition to this categorization during the first quarter of FY-90.

#### **G. JUSTIFICATION FOR NMPDS TO TRANSITION TO A DBMS**

NMPC-47 requested that an evaluation be performed on the need for converting the NMPDS data files to a database management system (DBMS). The study was performed by Oak Ridge National Laboratories through interviews with NMPC-47 personnel, contractor support personnel, and reading system life cycle documentation. Each of the four systems in NMPDS were developed individually, and therefore each one provides for its own input, databases, and output. Some of the systems use the same or similar data and so converting these systems to a DBMS can bring about a number of benefits not currently realized.

The study found the following benefits. Data could be evaluated on a NMPDS (or departmental) level instead of approaching each system for information needed. Data transfers and data redundancy produce a number of problems including data integrity, storage, and processing time. The second major benefit discovered by the study was that query and report capabilities would be expanded and improved in terms of data integration and availability. Allowing the users to generate their own requests would

improve system responsiveness to the user. The integrated data would allow for better decision support and modelling capabilities.

One of the possible problems with implementing a DBMS is that the present mode of software development is in using the Sage COBOL code generator. The decision must be made whether or not to continue this effort. Based on this decision, a DBMS product may be chosen which would interface the Sage code with the DBMS, or an interface must be developed between the two. The level of effort required to convert the files to a DBMS would be substantial, and so a commitment must be made to support the effort at all levels of personnel. It must be realized that the benefits from a DBMS outweigh the cost to convert the system [Ref. 2].

## H. CONCLUSION

The timing of NMPC-47 to reorganize the systems under more functional categories provides for an excellent opportunity to combine redundant functions and data in each of the systems by converting NMPDS to a database management system. The new capabilities provided by a DBMS are not currently represented in the Sage code, and so the conversion would require a new look at the applications programs. The current automated systems were designed providing the best support available; however, with the new developments in database software, these functions can be supported in completely different (and better) ways by the DBMS.

The Distribution Department is divided into ten divisions and each performs unique functions in support of personnel distribution. The divisions manipulate their own data but use corporate data as an information resource. This type of environment lends itself nicely to a distributed environment. Data owned by a division can be located near and changed by that division. Data that cannot be changed by the divisions, for example activity data or manning data, is updated through batch processing on a nightly or weekly basis. The primary exception to this rule is training reservation information. The detailers across divisions may reserve seats in training classes, and so this data can be changed by more than just one division. Synchronization of the seat reservations must be considered. This thesis assumes the decision to transition to a DBMS has been made, and investigates some models, architectures, and DBMS software to help NMPC-47 determine the appropriate choices for the NMPDS applications.

### **III. DISTRIBUTED METHODS**

#### **A. OVERVIEW**

Evaluation of distributed methods is done in this chapter by first comparing centralized systems with distributed systems. Some definitions of distributed systems are presented. Distributed Architectures, Process Handling Algorithms, and Data Handling Algorithms are discussed and evaluated. Conclusions are made in terms of management issues relevant to distributed systems.

##### **1. Centralized Versus Distributed Systems**

Traditionally organizations have developed and implemented automated support in a centralized environment. This allowed the software efforts to be coordinated through one organization to ensure consistency, standardization, and control. The central collection of data is better suited to top management requirements. In general, the cost of communications lines, hardware and support personnel is lower than that required for a distributed processing environment. Contingency options for remote sites, for example, do not cause major management concerns.

Many studies have shown that distributed processing and distributed data are the most efficient and practical ways to process information in terms of the future. A Distributing Computing Environment has been

defined as, "Any given set of computing systems, dispersed over a number of physical locations, which can be logically interconnected to form a harmonious and freely interacting computing environment." [Ref. 3] The distributed environment tends to mold to an organization's divisions better than a centralized one. The database and information processing, when divided by organization, lends itself better to the daily operations. Each division performs functions unique to that division. Data manipulated at the division level is usually not changed at a different level. Therefore, locally isolating data processing in an organization matches the distributed functions of the organization.

Distributed processing tends to benefit the user's needs and realize higher performance. The local databases are synchronized; there are no time delays in updates for data in the local databases. The number of application programs required for different geographic locations is minimized. Only the programs used by the local users are needed. Maintenance work by application programmers is reduced because the applications can be more specific to a small group of users' needs. For example, the applications in a centralized environment must meet all of the users' needs, so the applications will be general in order to fulfill all of the specifications. Often the application systems become very large, with a large number of programs to meet each user's needs. If the application programs are distributed, then only those required by the users at one site

would be located at that site. Application programmers are not required to find solutions that suit all users' requirements.

Improvements in data accuracy and better job performance by quick data collection and delivery are realized. Utilizing regional data enables decreasing communication costs. Overall reliability is improved by localizing system failures, and flexibility for upward scaling of database systems is improved. A distributed environment should have "data location transparency," that is, the applications are independent of the geographical data distribution [Ref. 4].

## **2. Distributed Definitions**

A major issue with system designers is defining a distributed database. The concept is relatively new, and different organizations have different definitions of what a distributed database means in their organization.

Infodata of Falls Church, VA, has two definitions of a distributed database. 1) Multilevel DBMS; the database and data are distributed to satisfy local requirements, then integrated into corporate databases by network interconnection. 2) Partitioned databases; there is no overlap in the local databases. Each center holds segments of data and the data is integrated by surveying segments [Ref. 5]. A Multilevel DBMS might be used in an organization where different levels of management require different amounts of data. A bank teller may require information on individual accounts located at that bank location. The bank manager may



require some of the same information on a less detailed level. The regional manager may require even less detailed information. There may be data located at each level and shared by each level of management. On the other hand, an organization where each division performs separate and unique functions and may operate with a partitioned database. There is no need for data to be shared. Most organizations would use some combination of the two methods.

Applied Data Research Inc., describes their view which allows users to connect dispersed data into a single information source. It ties together production data across mainframes, enables users to cross-reference dynamically, provides full read and update support, replication, partitioning, and transparent access to data at remote sites. The partitioning facility segments the database and it is put in multiple systems. Data with high availability requirements can be replicated in secondary or mirror image copies at multiple locations. A two phase commit protocol ensures replicas stay consistent with each other [Ref. 5].

Computer Corporation of America, Cambridge, MA, sees two primary views: 1) With multiple copies of a DBMS on an IBM-type mainframe, a single database can be broken up into pieces in many locations, a partitioned database. The system determines where relevant portions of the database are located. An access plan is worked out to retrieve data without user awareness of distributed data. There are multiple copies of the database distributed among sites. 2) The system can support

the redundant data through items stored on multiple computers which are updated automatically by the system. Loss of a machine does not block access to the data [Ref. 5].

An organization must agree upon a definition of what "distributed" will mean in their organization, especially in terms of data synchronization and system availability. Characterizing a distributed system according to particular attributes may help in determining the appropriate design of the system. Some attributes, which are mentioned above, and may affect the design of the system are: 1) division of labor in the organization, 2) level of data synchronization required, 3) system and node availability requirements, 4) machine failure contingency requirements, and 5) the level of distributed processing required by the applications. By placing emphasis on the appropriate attributes for a particular system, a common understanding of the system may be defined. A common definition will provide direction in terms of the system architecture, distributed algorithms, and communication protocols to be used. These decisions will determine the level of data synchronization and reactions to site or node failures.

## **B. EVALUATION OF DISTRIBUTED METHODS**

The massive amount of research being conducted in the "distributed database" field of study can cause confusion for users with limited knowledge in this area. Research institutes, universities, and private industry are all conducting research in slightly different areas of distributed

processing. Emphasis may be placed on hardware solutions, network communications, multi-model solutions, mathematical algorithms, protocols, or some combination of these areas to provide an accurate distributed system to the user. The above solutions may be broken down even further into more concentrated areas of research. The following section concentrates on the mathematical algorithms which can be translated into software solutions. These may be divided into the following areas: 1) Distributed Architectures, 2) Process Handling Algorithms, and 3) Data Handling Algorithms. Several distributed methods will be evaluated by placing them in one of the above three categories; however, these divisions are not clear cut. Some of the research efforts may cross into many of the above "emphasis" areas. A user organization, in particular the Database Administration personnel, must review these types of methods in order to determine the best solution for their users in terms of the attributes of their system.

### **1. Distributed Architectures**

A common method of specifying a distributed system is in terms of the architecture, or logical design, of the system. This is often done by dividing the system into some type of logical "sections" that will each manipulate the distributed system as defined by the organization. In other words, an architecture may be partitioned into layers, functional modules, or sub-networks. These logical layers may represent software or hardware divisions or both.

***a. Network-Oriented Operating System (NOS)***

One such effort is the Network-Oriented Operating System (NOS) described in [Ref. 6]. It was developed to promote resource sharing among participating host computers, and to better manage distributed communications. The NOS architecture was designed to coordinate parallel processing among mainframes, especially those used for research in the following areas: artificial intelligence, picture processing, scene analysis, and speech understanding. These types of machines require special peripheral devices such as analog to digital and digital to analog converters, image scanners, graphics displays, and color TV displays. These types of peripherals can devour a CPU's resources. The NOS was designed to include time sharing job scheduling, multiple parallel programming facilities, and the management of distributed databases to allow for parallel processing features. The concept driving the development of NOS was to provide an operating system designed exclusively for managing distributed processing through the network communications. By isolating the primary objectives of the operating system to meet these needs, a more efficient system could be produced.

***b. Network Access Processor (NAP)***

The Software Configuration for the Network Access Processor (NAP) is a different approach to managing a distributed system [Ref. 7]. The NAP contains program modules, named below, which perform the following functions:

- 1) The Monitor Program: task control, interruption control, and I/O device control are common programs performed by the Monitor.
- 2) The Network Control Program: services commands to run applications, controls interprocess communications, and performs analysis of network communications.
- 3) The Virtual Circuit Control Program: performs packet level protocols to provide virtual call services. The Virtual Circuit Control Program interfaces with the Network Control Program for call "request" or "terminate" signals, and interfaces with the Link Control Program for "send" and "receive" signals. This level is the interface between the logical and physical layers of communications.
- 4) The Link Control Program: link access procedures are handled by the Communication Control Unit hardware and the LCP. Unit hardware performs flag generation, abortion, CRC check, and bit stuffing while the LCP analyzes commands and responses and carries out error recovery.
- 5) The Gateway Program: provides access services and protocol conversion up to the packet level. Manages corresponding logical channels to support the virtual call.

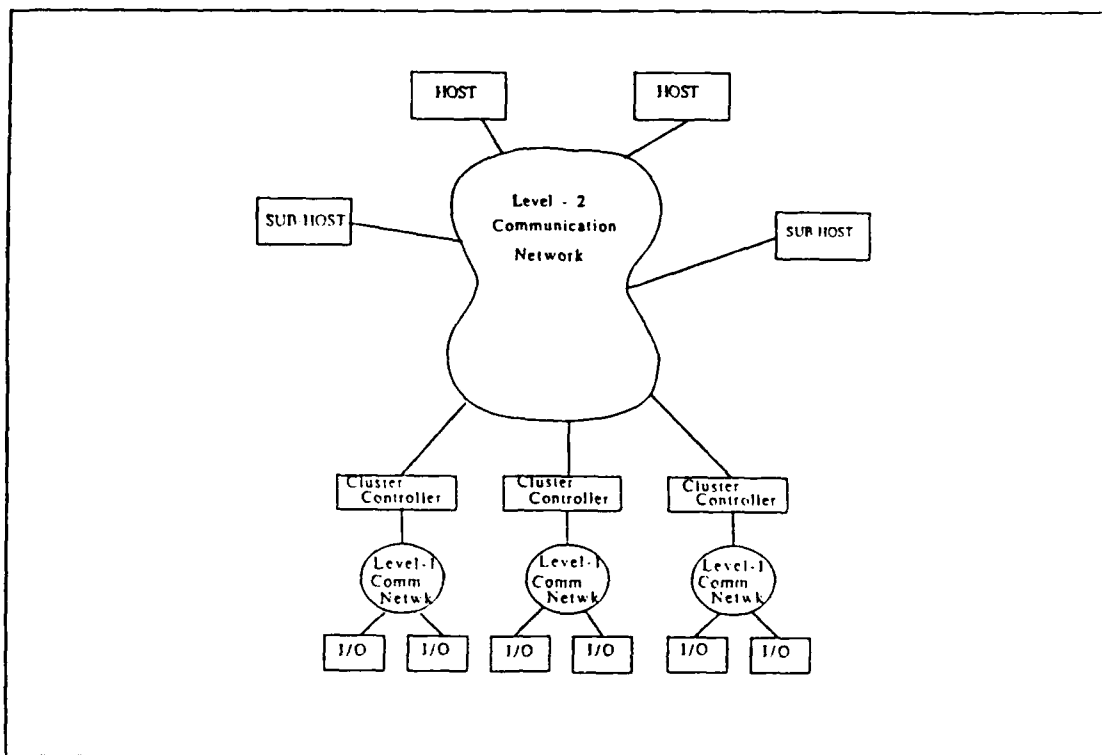
The layered structure of the software permits integration of new functions, for example, a Distributed Database Directory Service or Graphics Interface. This provides the logical communication facilities which interface to a network and support communications between various computer systems.

The functions required for communications can be optimally distributed in the network. The software configuration corresponds to the protocol of each layer. Only the layers affected by the new functions are changed. The system is not restricted to the original design.

### *c. In-House Configuration*

An "in-house" configuration with distributed intelligence, is described in [Ref. 8] as hierarchical levels of sub-networks which contain four types of nodes: I/O stations, cluster controllers, sub-hosts, and hosts. Figure 4 shows the architecture of the configuration. Level 1, the Serial I/O Loop, connects the cluster controllers and I/O stations and communicates in a master/slave relationship. Level 2, the Data Highway, interconnects the host, sub-hosts, and cluster controllers and uses peer coupling to communicate.

As nodes become more intelligent, for example microprocessors and memory in I/O stations, then the host, sub-host, and cluster controllers are released from dedicated I/O control functions. Dividing the network into the two logical levels provides the following benefits: divides the fast and slow data transmissions, clusters nodes based on access frequency, and divides the communication relationships. The features of the Serial I/O loop are that it provides easy attachment of new types of I/O stations, and easy addition or removal of stations. The Data Highway can provide high speed communication between its nodes, and make efficient use of distributed software and hardware resources. Limiting the communications



**Figure 4 Levels of Architecture**

requirements to a small geographic area allows emphasis on high-speed communications, a variety of communication means, and new communication techniques. The configuration was implemented and tested at Fujitsu laboratories in Kawasaki, Japan. This architecture provides excellent performance while being constructed economically, and Local Area Networks use this methodology.

#### ***d. Distributed System Environment***

Honeywell developed a Distributed System Environment, [Ref. 9], to support a wide range of configurations, interconnect different types of equipment, and provide transparency of the network environment to the application programmer and user. They defined a network architecture as

"a set of rules defining modularity, interfaces, and protocols by which functions communicate." The architecture must be compatible at all levels. The System Level defines relationships among components, while the Lower Level pertains to hardware/software features of particular equipment. System Level functions are administrative in nature and may be static or dynamic. Some typical control functions, which may be centralized or distributed are data base administration, monitoring and control of data communication networks, and the control of application development.

The Lower Level architectures include:

- 1) Information Processing: application related, performs functions normally executed by a general purpose computer.
- 2) Network Processing: provides control of data communication facilities.
- 3) Data base Management: control of data structures, access to stored data that form the data base.

With this architecture, functions can be selectively centralized or distributed. For example, central monitoring of status of equipment and communication facilities allows failures to be resolved from a central location.

*e. SHARD*

A final architecture to be reviewed in detail is SHARD: A System for Highly Available Replicated Data. SHARD is designed to support continued database operation in the face of communication failures and network partitions [Ref. 10]. SHARD uses timestamp-ordering to ensure eventual mutual consistency. Robust update, propagation protocol,



and mutual consistency mechanisms were developed to realize mutual consistency. Application-specific actions are triggered in the event of an inconsistency.

Replication of data allows greater availability in a distributed system, since sites may allow continued processing if one site fails. The requirement for multiple copies requires maintaining transaction serializability and consistency of the copies. SHARD permits users to continue accessing the system during a partition and updates are processed correctly after the partition is corrected. The application reads the "best data" at that time, updates are queued for transmission to other sites, and executed against local copies of the database at that site.

A reliable broadcast protocol ensures delivery of all updates to all sites within system. Site failures and Network Partitions cause delays in propagating updates, and different sites see the same updates in a different order. SHARD guarantees that all sites eventually will have the same database in spite of different orders for updates at different sites, by use of system-wide timestamps.

Traditional concurrency control resolves tradeoffs between different parts of the application database, depending on whether consistency or availability is more critical. SHARD is designed specifically to handle problems of partitioned networks, and the algorithms assume failures can occur at any time. When sites cannot communicate, one hopes the other has either failed or will not do anything that conflicts with the

site. If other sites did perform conflicting actions, the application takes appropriate steps to resolve the inconsistency.

A site is continually reconciling its database copy with new information arriving from other sites. There is never a need to detect a partition or reconnection or for sites to agree on who is in communication. This is important because it can be computationally expensive to do these things.

The System Design and Implementation are described in terms of the model of the database interaction and the system architecture.

- 1) Model of Database Interaction: SHARD maintains a database of state information and an application runs interactions against this database. Behavior of updates is more restrictive than the conventional model of transactions. An update can't perform input, output, or return information. Queries are not serializable with respect to all updates, but rather they see some subset of the total order defined for the updates. SHARD supports conditional updates which read information from the database state to determine how the database state changes. The effects of a given update will be known completely only when all preceding updates in the serialization order are known and have been executed.
- 2) System Architecture: "True" data is defined as the history of updates executed. Each site carries both an update history and a current view. Each site has a logical clock to assign unique update timestamps that

define total ordering. Each site contains three software modules which clearly divide responsibility: the Interactor, the Distributor, and the Checker:

- a) The Interactor presents data to the users and accepts updates.
- b) The Distributor implements a reliable broadcast protocol to ensure that all sites eventually see all updates.
- c) The Checker installs updates in the database and ensures mutual consistency using log transformations.

The transaction history and databases are located on stable secondary storage.

The configuration for the SHARD Prototype included a maximum of six sites. Larger configurations would benefit from work in these areas: partition replication, some notion of "priority" so when resources are limited the most important information is transmitted and processed first, an improved distributor process structure, and mechanisms to allow sites to be added and removed dynamically.

SHARD's systematic approach to achieving high database availability in spite of failures allows continued access to and updates of data. The system accepts some risk of non-serializable transaction execution. SHARD replaces the guarantee of strict consistency by an "eventual" consistency guarantee achieved with a combination of timestamp ordering and application's compensation mechanisms. SHARD's new features include support for database updates that are more complex such as

increment and conditional updates, and update propagation which takes advantage of intermediate sites. Shard allows an application to selectively trade off consistency and availability with the use of concurrency protocols. By allowing these protocols to be unblocked after an application specified amount of time, SHARD achieves a desired balance between the timeliness of response and the probability of inconsistency.

#### *f. Distributed Architecture Layers*

Distributed Architectures concentrate on the logical design of the system. In general, the systems are divided into layers or modules which are responsible for unique functions as defined by the system. The communication between the levels or partitions is critical to the success of the Distributed Architectures. The concept is not complicated for the Database Administrator to manipulate, since different functions are performed at different levels and changes can be isolated. The more difficult task is choosing an architecture appropriate for the system.

### **2. Process Handling Algorithms**

The algorithms discussed in this section manipulate processes within a distributed system. The software resulting from this type of research operates on a system level and manages the scheduling of processes. The algorithms require an in depth evaluation of the application processes to be run on the distributed system. Once a complete understanding of the processes is gained, applying one of these systems will allow distributed processing without scheduling conflicts.

### *a. Specification of Time Dependencies*

One such process handling algorithm was developed by Peter Ladkin, "Specification of Time Dependencies and Synthesis of Concurrent Processes," [Ref. 11]. This system uses interval calculus as a base to give very high level specifications of concurrent process protocols. These specifications are translated into a specification language, Concurrent REFINE, which allows identification of critical sections in the processes. Specification of time dependencies by means of intervals of time allows representation of process execution in a multiprogramming context. Interval calculus uses notions of duration and diameter for time intervals, so it is suitable for specifying real-time systems.

The simple case of specifying a time dependency is that in which a job retains control of the CPU until it terminates. All interrupts are disabled, and the job runs over a continuous period of time, a convex interval. In order to run several jobs on a single CPU at once, context switching of the processes is required. In such a case, interrupts must be enabled and there is no limit on the maximum number of jobs, requesting the CPU.

The Synthesis of Concurrent Process Specifications process produces executable synchronization skeletons for large mutual exclusion problems. Synchronization skeletons include statements incorporating synchronization primitives and assertions to be maintained as true during execution.

Refinement of the Specification is completed using Concurrent REFINE, and some constructs from a time system, TUS, a natural system of time units. REFINE allows combined specifications to be compiled. The refinement technique was developed to show how very high level specifications of concurrent systems are transformed into specifications accommodated in the existing system.

The general strategy is to identify critical sections and assertions for the "guard predicates," and to assemble synchronization primitives into the skeleton for processing. The technique involves the following steps:

- 1) Introduce auxiliary predicates.
- 2) Construct a set of process executions with which a process is mutually exclusive.
- 3) Formally decompose critical sections into "request" and "execute" parts.
- 4) Construct conditions which assert that "a guard predicate is true only when it is safe to execute", and
- 5) Order the guarded statements within the calling process.

If two processes may not run concurrently, they are specified as disjoint from each other. Critical sections may be identified by constructing for each procedure the set of processes disjoint from it.

The refinement process demonstrates automatic identification of critical sections, automatic ordering of synchronized primitives within

processes, and formalization in the interval calculus. One of the limitations of this methodology is that one cannot reduce all synchronization problems by this process, in particular those which are recursively undecidable.

Interval calculus is a powerful tool for the specification and refinement of concurrent process configurations. Unions of Convex Intervals Calculus is expressive, allowing flexible, extensible specifications of time dependencies.

#### *b. Distributed Scheduling*

Another version of a Process Handling Algorithm is called "Distributed Scheduling Using Bidding and Focused Addressing." [Ref. 12] This algorithm concentrates on scheduling problems in real-time computer systems by finding available nodes to run processes. It uses a heuristic distributed method to schedule tasks, while considering both tasks with deadlines and task resource requirements. The algorithm is a flexible scheduling algorithm for loosely coupled distributed systems.

The system model that was developed contained three major components: resources, tasks, and nodes. The resources are active if they have processing power, passive if they don't. Tasks have characteristics attached to them which include: worst case computation time, a deadline, and resource requirements. At each node, there is a scheduling component which has: 1) a local scheduler which decides if a node can guarantee a task, 2) a bidder where an unguaranteed task is sent to and combines bidding and focused addressing to find a new node to schedule the task, 3)

a dispatcher which actually schedules the guaranteed tasks, and 4) a node surplus manager which periodically calculates node surplus.

There is a scheduling scheme which coordinates the scheduling steps in the system. The first step is to invoke the local scheduler to try to guarantee the new task. If it can be guaranteed, it is put into that node's schedule; otherwise it is sent to a bidder. The next step is to use focused addressing to select a focused node with sufficient surplus. Focused addressing estimates surplus of the nodes based on partial knowledge about the other nodes in the system. If there is a node with sufficient surplus, it is a focused node, and the task is guaranteed. If bidding is required, a request for bid message is sent to a subset of other nodes. The nodes receive the request for bid and calculate the bid. Results are sent either to the focused node or the original node. When the task is sent to the focused node, it invokes the local scheduler. If the node can accommodate the task, then the bids are ignored; otherwise the best bid is selected. If there is no focused node, then the original sends the task to the best bidder. A task cannot be guaranteed if there is no focused node and no best bidder.

The scheduling tasks required by a node are as follows: the scheduling component guarantees the task, and a new schedule is created, the new schedule replaces the old one and determines a start time, and if the task is not guaranteed then the schedule is not changed.

A heuristic function determines a feasible schedule for placing a task in a schedule. Distributed scheduling occurs when a task is not



guaranteed in the first node. Node surplus is computed. State information is used to help other nodes decide scheduling. An approximation of the node state is made to guarantee a task. The surplus is periodically calculated, sorted and stored in another node.

An evaluation of the algorithm, using a simulation model, was done. Six nodes were in the model. Each node had five resources, two active, three passive, and there was one periodic task per node. In the simulation results, the algorithm performed at an almost optimal rate.

Determining a perfect schedule for real-time task scheduling is impractical. A heuristic approach to perform on-line scheduling of tasks is needed. Heuristic functions determine a schedule for tasks executing on a node. Nodes cooperate through a combination of bidding and focused addressing. Despite communication overhead, the algorithm performs close to one with perfect state information.

### *c. Task Response Time Model*

A distributed method which handles its processes in terms of task response time uses a control-flow graph to represent the logical structure and relationships among modules. "Task Response Time Model and its Applications for Real-Time Distributed Processing Systems," [Ref. 13], introduces an analytical model to estimate task response time for loosely coupled systems. Application tasks are partitioned into several sub-tasks assigned to a set of processors for processing. Upon completion of a task, a module sends messages to enable the next modules to execute. It

may also send messages to update shared data files. Modules may interact on the same system, intermodule communication (IMC), or between different computers, interprocessor communications (IPC). It is the interprocessor communications that require extra processing such as communication protocols, management of distributed data files, and network delays.

Distributed systems require better solutions to overhead caused by modules on different computers sharing data. This is complicated by design issues such as module and file assignment, scheduling policy, and database management algorithms. Existing systems have used the trial and error approach, so a systematic methodology for designing distributed systems is required. Therefore, the development of an analytical model to estimate response time which can be used for exploring design issues has been completed.

A common model of a distributed system is represented by a queuing network with servers, customers, and arrivals. A FORK represents a module which enables more than one module. When there are several modules which must complete execution before a succeeding module is executed, the situation is referred to as a JOIN. The routing scheme in a queuing network is inadequate to represent these logical relationships (ie: FORK and JOIN), and so a specific "Task Response Time" model has been designed. The model contains two submodels: "Module Response Time

Model" and "Weighted Control-flow Graph Model" which are used to represent the algorithm.

The Module Response Time Model estimates the response time of each module in a system. The response time extends from the time of invocation to the completion of execution. It includes waiting time, execution time, output IPC time, and input IPC time. Each computer (or processor) can be considered a separate queuing system. If several modules are being invoked, it is called "bulk module invocation." Module arrivals in the system are independent of each other and interarrival times are Poisson distributed. Bulk invocation modules are invoked at the same time and the operating system schedules an execution sequence based on resource requirements of the modules. So, the average response time for a module is the average module bulk waiting time and the sum of execution times of the individual modules.

The Weighted Control-Flow Graph represents the intermodule relationships and the module response times combined. The response times are mapped onto the control-flow graph as arc weights. The task response time can be estimated from this model.

Control-flow subgraphs can be of four types: sequential thread, And-Fork to And-Join, Or-Fork to Or-Join, and loop. These are basic logical relationships, and there may be one or more of these subgraphs contained in a task control-flow graph. A sequential thread is a series of modules where each module has a single successor. The total response

time is the sum of all arc weights. An And-Fork to And-Join subgraph represents a module which simultaneously enables several modules, all modules complete execution, and a final module is enabled. The response time of the subgraph is the maximum time of the individual modules' paths. The Or-Fork to Or-Join module enables one of its succeeding modules based on some branching logic. The response time is the weighted response time of all threads. A loop subgraph represents a process of repeatedly invoking a set of modules. The response time is the average number of times a loop is executed multiplied by the time required to execute a single loop.

A generalized model is introduced to compute response times for dependent module invocations. The graph is partitioned so that the modules of each subgraph are allocated to the same computer. The invocation of the modules are dependent on each other, and the dependencies at forks and joins complicates the model. This approach provides more accurate response time when modules form a long sequential thread.

The Task Response Time Model can be used to study the effect of response time on issues such as module assignment and precedence relationships, scheduling disciplines, and database management algorithms. Distributed systems require protocols to ensure internal and mutual data consistency for simultaneous access of replicated data files.

The results of using the model provide insight to performance and overhead of concurrency control algorithms.

The approach of this model considers queuing effects, interconnection network delays and logical relationships and therefore provides accurate time predictions. It can be used to study the module assignment problem and the effect of precedence relationships among modules. In addition, it can be used to study design issues such as module scheduling policy and database management algorithms. The model serves as a valuable tool for systematic planning and designing of distributed processing systems.

#### *d. Evaluation of Process Handling Algorithms*

Process Handling Algorithms are designed to concentrate on efficient processing in a distributed system. Each of the above algorithms could be used in a different type of distributed system. For example, the first method, Specification of Time Dependencies, is suited for a system that contains processes that depend on other processes, or a tightly coupled distributed system. The Distributed Scheduling algorithm, designed for a loosely coupled machine, is geared toward systems where the processes are competing for resources but are not dependent upon each other. This algorithm might be used in a system where there are many independent processes running, and some of the processors are overloaded. The final algorithm, the Task Response Time Model, might be used where response time is of utmost importance, for example a weapons system or navigation

system. Depending on the type of processing in a distributed system, an appropriate processing algorithm can be selected.

### **3. Data Handling Algorithms**

A major consideration in a distributed system is the data handling procedures. The increasing amount of automated information processing will produce databases that exceed the physical limitations of centralized systems. Algorithms can be designed to efficiently manage the distribution of data in the system. It is important to first determine priorities in terms of data synchronization and integrity, communications delays, and response time.

Peter Apers presents an algorithm in "Data Allocation in Distributed Database Systems," [Ref. 14] which provides a solution to allocating data to sites in a distributed database system. His view is that allocating data in the "distributed file problem" is not the same as in a distributed database. Data access is more complicated. A relation may be split, either horizontally and grouped by tuples, or vertically and grouped by attributes. If resulting fragments are located at different sites, then the database is "partitioned." If copies of relations are placed at several sites, the database is "replicated." These features of a distributed database, partitioning and replication, can present both benefits and problems that do not exist in a centralized processing system.

A typical problem with distributed query processing is that some algorithms require the DBMS to supply the fragments and ensure a

consistent view of the database is being provided. Another problem is determining processing schedules. Parallel processing and computation should be factors in an algorithm which determines schedules for queries where a certain cost function is minimized. If a query requires a "join" or a "fork", the algorithm must know something about the execution in order to correctly represent it.

The cost of an allocation is computed by a processing schedules graph. A graph is constructed with 1) Physical site and virtual site nodes, 2) Edges that represent transmissions of the processing schedules, and 3) the operations of the processing schedules to the operations sets of the sites. The graph contains information necessary to check constraints such as bandwidth, CPU-utilization, and node availability to determine the cost of queries and updates. The goal is to obtain a completely specified allocation by manipulating partially specified allocations such that a given cost function is minimized. Cost functions can be minimized by utilizing parallel computations. The forking process is used to start a parallel computation. The forking graph is a subset of a processing schedule graph. A good heuristic to minimize response time is to allow for as much parallelism as possible. Relations are split into fragments based on user queries, and further splitting horizontally will enhance parallelism.

The costs of many different allocations must be compared to compute optimal allocations. Static processing schedules avoid recomputing schedules by starting with an initial allocation and adjusting the graph. An

initial allocation is, for each query and update, a copy of a fragment created and placed in its own virtual site. Schedules are then computed and placed in the graph.

Techniques such as branch-and-bound or heuristics are used to search the large solution space for determining data allocations to minimize total transmission cost. These techniques construct decision trees. A node is identified by the path from the root to that node. The edge corresponds to a decision taken about data allocation. A subset, belonging to a leaf, contains all completely specified allocations that satisfy the partially specified allocations defined by decisions taken to reach that leaf. The cost of a subset is the minimum cost among all solutions in the subset. The cost-estimator is the sum of two components: 1) the cost caused by decisions to reach the partially specified allocation from the initial allocation, and 2) the estimate of cost caused by decisions that still have to be taken to reach a completely specified allocation with the least cost.

The algorithm is as follows. For each iteration, a leaf with the smallest cost estimator is expanded. Expanding a leaf means unite one of the virtual sites with each of the physical sites. Continue until a leaf with a subset which contains only one completely specified allocation is chosen as the result.

The cost estimator of a subset underestimates the cost of the subset. Otherwise the search deteriorates into an exhaustive search. If each virtual



site is connected with only one physical site, the cost estimator could be determined based on transmissions between physical sites.

A Heuristic Data Allocation algorithm was evaluated and proceeds as follows: start from an initial solution and locally optimize until no improvements are possible. When several are possible, the one that decreases the cost function most is chosen. Algorithms using this technique are called greedy.

The above tools may be used by one or a group of database administrators. The algorithms can be applied to centralized data allocation or distributed allocation. In Centralized Data allocation, allocation of all the data is considered at the same time. If either one data base administrator or one central data base management system is allowed to change the existing allocation, then the system is considered centralized. All queries or updates are used to determine fragments and to compute a completely specified allocation that minimizes a particular cost function. Algorithms can be used for minimizing total transmission cost. The allocation obtained will be implemented by the data base administrator who can dictate an allocation to local data base management systems.

Decentralized, or Distributed, Data Allocation exists when data is owned by different data base administrators. or the distributed database is a collection of databases owned by different parties. No central organization exists that dictates the allocation of data. Therefore the database management systems of the sites should cooperate with each other and try

to determine an optimal allocation of data required by users at their sites. Users at a site who share the same view of data can request their local system change the allocation to minimize a certain cost function. An advantage is the natural partition of the general data allocation problem into a number of smaller ones, solved more easily. Another advantage is that data allocation can change more or less continuously through time. If access patterns change, the database administrator at that site simply determines a new allocation for them. The overall cost of the decentralized approach may be higher because only smaller portions of the scheduling graph are considered at a time.

This model was introduced to compute the cost of allocations for various cost functions. It is suitable in both branch-and-bound and heuristic algorithms. A method for determining unit allocation by splitting relations in the conceptual schema base on queries and updates is presented. A framework was discussed for managing allocations in a distributed database with one or more database administrators.

This type of algorithm would be ideal in a large distributed system where each of the sites requires data from most of, or all of, the other sites. For example, an inventory control database system, where many parts were supplied by many manufacturers, and these parts were distributed to locations in many different areas. If no consideration is given to the data allocation, eventually there will be very limited communications between

sites. The network communication would be overloaded trying to transfer tuples to all sites on each occurrence of an update.

## **C. CONCLUSIONS**

Each of the distributed methods described in this chapter approaches the distributed processing, or distributed database management problems, from a slightly different angle. The distributed architectures view the system from a logical or organizational point of view. It is necessary to define the roles that will take place in the management of the data and processes. The responsibility of these roles must also be divided and assigned to the appropriate modules or layers defined by the architecture.

### **1. Management Issues**

The success of a distributed system involves not only the configuration of the software and hardware of the system, but also on appropriate management techniques in the emphasis on appropriate issues such as performance and integration.

#### ***a. Performance Tuning***

The lowest levels of management in a distributed system are the process and data handling routines. Some type of organization and systematic approach is required in order to efficiently operate a distributed system. These are the algorithms which must closely match the user organization's needs. These algorithms can be and should be tuned as the system is being used in order to meet changing requirements.

### *b. Integration*

A primary problem in today's distributed environments is to connect the variety of equipment from different vendors. Equipment provided by one vendor doesn't understand the management protocol used by another vendor. Neither reports operational or status information in the same way.

Some vendors offer management capabilities with hardware, or provide software to tie it all together, but some organizations cannot always take the single source route. Common carriers do not offer freedom of choice. Independent carriers do not provide adequate coordination for customers' service. Communication standards are lacking for the micro-to-mainframe world, although Open Systems Integration (OSI) is helping clear this sore spot. Network Management requires good communication and coordinated efforts between staff, users, and nontechnical management. Effective network management combines people, philosophy, and technology. New network monitoring services are being offered to companies which are all encompassing and include such efforts as: isolating problems, rerouting traffic around failures, and informing affected vendors and carriers.

The OSI efforts are ongoing, and a working group met in Tokyo to refine network management standards [Ref. 15]. This includes standards of all resources, including hosts, in a mixed vendor setting. OSI standards are hierarchical in nature and were developed in a top-down

manner. OSI is strictly concerned with interconnection between systems that allow exchange of management information between remote devices and centralized operations staff. Specific standards apply to the five areas of management defined as: 1) configuration, 2) fault handling, 3) security, 4) performance, and 5) accounting. This model of management, OSI, is meant to bridge the gap between abstract standards and the real world of customer needs. OSI allows independence from vendors, and will eventually enable networks to be easily tied together.

## **2. NMPC Requirements**

If NMPC evolves to a distributed environment using workstations, the primary concern will not be data synchronization. In general, the detailers have their own personnel, or constituents, to work with and assign to duty stations, and currently work with their own "personal database." Likewise, the placement officers work with their own activities, and use the information in the activity database which concerns their activities. If they have a need to see other information, it is not to make changes but merely to view the data. This viewing data can be updated in the master files nightly from each of the workstations. Failure of one workstation would affect no other users. That one user would merely need to find another operating workstation to use.

Training reservations, on the other hand, would require on-line updates throughout the distributed system, especially when there are detailers competing for use of those seats. Therefore, some type of

communication algorithm will need to be set up to coordinate the sharing of this data. A close evaluation of the data should be done in order to determine the data which must be synchronized. For example, the training data may be partitioned based on the kind of training. Each location may have one node responsible for allocating its seats.

The primary concern will be process handling especially at the beginning and end of each workday. As users log on, the central machines will be very busy downloading to the user his portion of the database, or the data he requires to do his job. At the completion of each workday, users are logging off of their workstations and uploading their databases back to the centralized databases. The processors of the machines will be very busy at these times of day, and efficient process handling routines will be necessary. The processing throughout the night will involve reconciling the updated copies of the data with the master databases. Jobs are run that create output to a variety of different organizations. These jobs must be evaluated to utilize the maximum potential of the CPUs. Currently the scheduling of the jobs takes most of the night; however, only about 20% of the CPU is being utilized, [Ref. 16]. Therefore, it is apparent that parallel execution of the jobs is required to prevent an eventual system backlog.

## **IV. OBJECT-ORIENTED DBMSs**

### **A. INTRODUCTION**

Resistance to change has been a major problem over the centuries for those few pioneers willing to explore the unknown. It is human nature to remain in a state in which we are comfortable, and when we try to introduce new "foreign" ideas there is often great opposition. This is the case with some of the advanced programming language concepts now being researched. Many programmers who are comfortable in their niches using COBOL, FORTRAN, Pascal, C and other traditional languages are unwilling to explore the unbounded possibilities available to them through concepts such as object-oriented programming.

In the academic world, people tend to desire searching out the unknown. However, this is unfortunately the exception rather than the rule. As the world of computers advances with each day, the members of that world, hardware and software, must also advance to keep in step. In the past 20 years, this has not been the case. Computer hardware designs have grown quickly so that size and speed improvements are barely implemented before new designs have been completed. Unfortunately, software design improvements show a slow start next to their hardware counterparts. End users are expecting more solutions and faster computers. Eventually, there

will be a limit reached for the hardware improvements. There will be two directions to go once this limit is reached: 1) parallel (distributed) architectures, and 2) better software.

A primary area of interest in the research community is object-oriented programming. The concept of object-oriented programming is to define all items in the construction of a program as objects. An object controls some local data, which can be manipulated from the outside only via the methods defined by the object. When an object receives a message, which requests some service and contains any needed input data, the object selects a method to provide that service. As more objects are defined, the level of abstraction through which the program is being viewed becomes higher, because the new objects are defined in terms of lower level objects. So the software gets closer to the level of specification and therefore closer to the statement of the problem. Programs should then be easier to understand and more correct. The software can be reused, because each object is a separate and complete entity of software. This encapsulation of code allows programs to be developed quickly because the tedious details have already been dealt with and solved. Objects can be put together to form larger objects. Brad J. Cox compares object-oriented programming to "Eli Whitney's interchangeable part innovation." Both redefine the unit of modularity and allow the production of subcomponents. The subcomponents may be controlled by standards and can be interchanged across different products. Subcomponents of software programs have been



termed "Software Integrated Circuits" or just "Software ICs" to emphasize the similarity to their hardware counterparts [Ref. 17]. These software ICs can be placed in or removed from programs as the developer sees fit on any machine. The hardware that the programs run on is immaterial, because the ICs are fully portable; they are designed to perform specific functions. In the face of an everchanging world, flexibility is a key factor.

The ability to reuse objects that have already been developed allows programmers to continue to build upon previously completed work. In general, object-oriented languages are not difficult to learn; they tend to be relatively simple. Programmers can build to higher levels of abstraction quickly. Learning the numerous "Software ICs" already present in the object library in *SmallTalk-80*, for example, may be more intensive than for a traditional programming language. However, it is exactly this use of the object library that makes the object orientated programming method so valuable [Ref. 17].

Building programs at a higher level of abstraction, using the object library, produces better programs. The programmer can become more concerned with the higher level logical problems and not worry about errors caused through performing the menial tasks. It is quite often the case that a programmer spends hours debugging a program just to find a very simple mistake, say a missing definition, that was causing some very major problems. The use of Software ICs in object-oriented programming can reduce this type of problem while allowing the programmer to concentrate

on higher level issues. The benefits provided by object-oriented programming can easily be transferred into object-oriented database management systems.

Encapsulation is realized by a collection of compartmentalized objects that communicate only by sending messages. The objects act as "black boxes" of code, and the implementation inside the box is restricted from the user's view. The methods used in accessing objects allow manipulation of data before the user sees it. The object determines the method used to access the data based on the message received. In an object-oriented database, the user does not need to be concerned with the access method an object may use to produce the desired information. The object selects the appropriate method based on the query. There is no need for an end-user to learn a language specific to a database model, like a relational query language.

## **B. FEATURES OF OBJECT-ORIENTED DBMSs**

The features of object-oriented database management systems provide new and useful tools to the database environment. The use of message passing and data manipulation within an object allow processing to occur out of sight from the user. The processing is prompted by the message input received by an object. This provides automatic triggering capabilities within an object-oriented environment. Such a feature can produce some very useful applications. For example, in the context of a the Naval

Military Personnel Distribution System, if certain actions are completed, like the approval of a set of orders, then a follow-on action, the costing of those orders, may be automatically triggered. However, objects are designed with a series of methods, or actions, to perform depending upon the input received; so if the orders were disapproved, the set of orders may fall into a separate class of objects, and a completely different set of actions may be triggered by the object. If the orders were approved, then upon completion of the costing, the writing of the orders may be automatically triggered. These steps would not require input from the user, and so would free the user to perform other activities. In a conventional database management system input from the user is used to start each process. Implementing features such as the above in a traditional database management system may require a multitude of "case" statements or conditional statements. Such features would severely strain most DBMSs on the market today.

Concurrency and distributed systems are additional features of object-oriented programming that provide new tools. Objects can run concurrently. They communicate by passing messages and are like independent automata working in parallel and providing data where needed. The concurrency in object-oriented systems can provide such features as tracking software design history or version control. In many design and engineering applications, it is necessary to keep track of the development history. A development history often consists of a branching set of development paths, where each development path consists of a sequence of compatible

refinements. Branches appear in cases where alternative designs were considered or different configurations of a product were developed for different customers. Similar structures appear in planning activities and "what-if" analyses in systems modeling or games theory.

In object-oriented database systems, generic methods for version control can be provided as methods for a generic class "VersionControlEntity". These methods can be inherited in any application schema simply by declaring the classes of application objects needing version control as subclasses of the general object class "VersionControlledEntity". This has several advantages:

- 1) The version control code can be defined once and used many times,
- 2) The overhead of a version control mechanism is incurred only for those object types that need it, under the control of the application schema defined.
- 3) Improvements in data structures and algorithms for version control can be made by the system administrator without affecting any of the application schemas or programs.

The object-oriented concept is very similar to the concept of distributed processing, defined as many machines working independently and communicating data needed through interfaces. The design of distributed systems becomes easier to define because the design is similar to the design in an object-oriented environment. The level of abstraction in the database management system is close to the architecture of the system, and higher

level solutions can be designed which are closer to the specifications of the actual problem.

Extensibility is a feature of object-oriented programming that is not easily represented by traditional systems. The ability to define new extensions in a database that do not fall under the traditional data processing categories can be a difficult task. Recursive definitions and recursive query processing are not features found in traditional database models. The ability to define a set based on a subset is a natural representation in an object-oriented database. Recursive processing and specialized access patterns can be realized using the methods associated with objects. A common data structure is a graph consisting of nodes and edges. "Traverse the edges to find the shortest path" might be a typical query on this type of data structure. Object-oriented database features allow this to occur naturally, as objects may easily search their sub-components for the shortest path, and so on recursively. It is possible to define a graph in, say, a relational database model. However, edges and nodes may be spread in tuples across the system. The query may require some extensive database operations in order to gather the data required to solve the problem which are difficult or impossible to express in conventional database query languages. Applications which are recursive in nature can be easily specified in an object-oriented database [Ref. 18], and specialized data representations can be defined to make them more efficient.

The ability to reuse software in object-oriented databases is very similar to the use of "packages" in Ada. Packages can be defined in generic terms, and then be reused by a variety of more specific applications by creating instances with different values for the generic parameters. The primary difference between the concepts is that object-oriented systems allow the run time binding of data types, based upon the message received, and so can work with loosely coupled collections that hold objects of different types. Object orientation allows defining new types at run time by reusing some existing type. Ada expects binding to a data type at compile time. Ada was designed as a strongly typed language for use with embedded real-time systems [Ref. 17]. The ability to define generic or parameterized object classes is essential for an object-oriented database system to realize the full benefits of code reusability. This allows generic standard object classes to be tailored to particular applications by supplying particular values for parameters and then inheriting the tailored facilities in particular applications. For example, a generic class can provide a general purpose method for displaying a structure chart, where the relation defining the structure, the procedure for extracting the label from each node, and the shape of the icon used in the display are generic parameters. Such a method can be instantiated in different ways, for example to display the subcomponent hierarchy of a mechanical assembly, or the uses relation for the modules in a software design, or an organization chart for a corporation.

Specific solutions can be developed using traditional databases and the results are very specific applications. General models and solutions are not enough to solve more complicated problems, but specific solutions are not able to be reused by other applications. The ability to parameterize generic solutions allows them to be adapted to specific situations.

Another important feature provided by object-oriented database systems is the ability to add new data types to the system. Geographic, spatial, or temporal representations of data types may be defined by defining new object classes in an object-orientated system. These are data types not supported by traditional database systems. For example, the three dimensional representation of a geographic area on a radar system may not be easily represented in a relational DBMS. Determining positions in that geographic space would require extensive computations. Then, representation of the data values would be converted to some form the end-user could understand. An object-oriented database, on the other hand, can send such a request to the objects it contains, and the object can determine methods required to produce the results to the user. The concepts are represented as objects within the geographic object class. Concern with the appropriate numeric processing, for display purposes would have already been resolved when the lower level objects had been defined. This feature allows a database to expand beyond the usual capabilities available.

The object-oriented database can treat different relations as specialized classes of relation objects. This would allow efficient support for

specialized features such as security, performance, and specialized methods. An object may select the portion of the database that a certain user is allowed to see. This automatic partitioning of the database makes data security issues easier to address. Similarly, an object may also choose a specific implementation of an application. By allowing the objects to choose the implementation, a more efficient version or more specialized version may be chosen depending on the needs of the user. Specialized methods may be made available only to those requests with authority to do so. The object may choose the appropriate method for those requests with access, and execute them. These methods may have been designed for a certain group of users.

### **C. EVALUATION OF OBJECT-ORIENTED SYSTEMS**

In this section, an existing object-oriented database management system, PROBE, is discussed. The research supporting PROBE is being conducted by a division of Computer Corporation of America which is now owned by Xerox. An algorithm for determining unique identifiers for objects in a distributed system is also discussed.

#### **1. PROBE: An Object-Oriented, Extensible Database System**

Support of spatial or temporal data processing is difficult, since each application has its own notion of space and time. The approach in PROBE is to define a mathematical abstraction of spatial and temporal objects. PROBE provides built-in support for the general concepts, so



different types of applications can be mixed. Recursive queries occur naturally in some problems. Adding general recursion to a query language which does not support recursion is very difficult. PROBE supports traversal recursion and self referencing queries through its support of recursive definitions in the object-oriented database.

Most database systems do not provide facilities for extensions to their systems code. PROBE supports extensibility. Major components of the system are: 1) The Database System Kernel: a query processor designed to manipulate objects of arbitrary types, and 2) A Collection of abstract data types or object classes: these classes specify the representation of objects of a new type and provide operations to manipulate them. It is necessary to specify the division of labor between the database kernel and object classes. The kernel should handle sets of generic objects. The object classes should handle individual objects of specialized types. The Application Specialists supply the object classes and will not have to consider database system implementation issues. The Database Implementors deal with the interface between system and object classes, and will not have to consider application specific issues [Ref. 19].

Extensibility is supported with the notion of generalization. At the root is the most general type, ENTITY, and from the root object types with more specific semantics are defined. These object types are defined in terms of the object, or entity, previously defined. For example, Body-of-Water may be a type of ENTITY. The user has defined an object of type

Body-of-Water. Other objects may be defined in terms of Body-of-Water. Lake, Pond and Ocean may be examples of objects belonging to the Body-of-Water class. A query, ChemicalBreakdown may be submitted, and depending on the class of type Body-Of-Water, the object selects the appropriate method to produce the results of a chemical breakdown. The object decides at run time the method chosen to execute. The user can define not only nontraditional data types, but he may also define nontraditional operations to execute with the extension. This extensibility starts with the most general case, ENTITY.

The PROBE Data Model's (PDM's) basic constructs are entities, functions, relationships among entities, and operations on entities. Entities are real-world objects, and functions represent properties of entities. Functions and entities are manipulated by the PDM algebra, an enhanced relational algebra used to manipulate the lower level requests natural to relational type queries. The system includes the notion of an entity, the ability to manipulate entities, support for computed functions, and support for spatial and recursive queries. It allows for the definition of new entities and functions by the end-user through the object-oriented extension.

In order to add an object class to the model, the database implementor constructs a specialized database system. An interface between the extensible database system and the object classes is well defined, so the construction is much simpler than modification of a conventional database system. If a modification to a relation occurred in a Relational DBMS

(RDBMS), adding some new attributes to a relation, then everywhere in the application that tuple is referenced would have to be changed. In an object-oriented system, such a change would only require adding a new method to an object class, and none of the existing applications programs would be affected by the change, other than the new application added to use the new information added by the schema extension.

The PROBE database system meets the needs of nontraditional application areas not met by traditional database systems. It is possible to represent complex arbitrary data types, and specialized operations that can be difficult or impossible to specify in traditional database systems. PROBE supports extensibility by incorporating arbitrary object classes.

The PROBE Data Model contains objects and functions and is extensible because new objects can be easily created. This approach is supported by architecture where the database system kernel is concerned with sets of objects, while object classes are concerned with application specific details. The breadboard implementation of PROBE includes the Data Model and query language. Object classes were added to support a simple geographic application and three dimensional data to support an air traffic control application [Ref. 18].

PROBE is an excellent database system for any data processing application not considered traditional data processing. The ability to extend the application to meet any user's needs makes PROBE a system with unbounded possibilities. Even traditional database applications can benefit

from an object-oriented DBMS because of the additional features provided such as reusability of software, concurrency attributes, automatic triggering mechanisms, and information hiding.

## **2. Robust Generation of Unique Identifiers**

Managing objects in a distributed environment can be as complicated as managing data in a distributed environment. Algorithms can be designed to efficiently manage the objects. The portability of objects in a distributed system can cause identification problems. The method described in "Robust Generation of Unique Identifiers in Distributed Systems," [Ref. 20], addresses the problem of generating unique identifiers for objects in a distributed system. Objects are location independent and each object is assigned a permanent identifier. The object can then move from one node to another and the object identifier does not change. Nodes in a large distributed system can be created and destroyed on a regular basis, and so generating node identifiers can be a complicated task. Because objects are portable, the identifier must be unique regardless of time and location of the objects' creation.

The objectives of the method are to generate globally unique identifiers, eliminate the need for a vote by nodes, minimize storage space required for an identifier, and minimize communication between nodes. An existing node runs an id generation algorithm which has access to its own unique identifier. It is responsible for assigning an id for a new node and is called the parent. The system starts out with a preallocated number of nodes

whose identifiers are manually assigned. When a new node is created, an identifier is generated and its existence is recorded into the system catalog.

There are four methods for generating identifiers: the centralized method, a partially distributed method, a completely distributed method with minimum communication, and a completely distributed method with modest communication and balanced use of name space. A centralized id server is the parent of all other nodes and produces identifiers that are optimal with respect to identifier length. It is unreliable because it is dependent on that one node. The partially distributed nodes reduce the sensitivity to failure by replicating the central server node. Each identifier server controls its own name space and operates independently of the others. It is vulnerable to failures, such as a network failure, but would perform well in architectures with communications redundancy. The minimum communication method is fully distributed and requires only the identifier part of the parent node as input. Any node can act as a server. The system is not vulnerable to network partitioning failures since a server is resident at each node. The method uses the genealogy concept for identifying objects, which creates a family tree of identifiers. This can lead to a long node identifier and is expensive in storage, but it is reliable in terms of node and link failures.

The final method addressed in this paper is a balanced robust identifier generation. The length of the node identifier is reduced by slightly increasing communication. The fully distributed method is used in

conjunction with a facility for tree balancing. This version most effectively reduces communication costs and susceptibility to network failures while at the same time reduces storage required for storing the unique identifiers.

Reusing identifiers is a necessity, since the size of preallocated identifiers to accommodate a maximum number of nodes is very large. A free list is associated with each node when sibling nodes are destroyed. Then it is checked first when new nodes are created. The free list size must be controlled in large systems, and can be metered to measure optimal system performance.

Four methods for generating unique identifiers were presented. The last version makes use of methods which generate optimal identifier length while minimizing the effect of network partitioning failures. Consideration of object allocation in an object oriented database system is required in order to maintain an efficient system which can respond to the users needs. The system must at the same time be able to respond to node failures in order to allow the applications to continue processing while minimizing the system resources, memory space, used.

#### **D. CONCLUSION**

With the demand for more accurate, faster, more versatile software, today's programmer will be forced to approach new ways of programming. The object-oriented concept of Software ICs applied to database management functions will enable database software to be reused, and

eliminate constantly repeating code, especially for menial tasks. The extensions made available by object-oriented DBMSs provide unlimited possibilities in the types of application problems that can now be solved through database management.

Traditional database systems are suitable for applications that can be easily described in terms of tables containing numbers and short text strings, especially if there are only a few types of tables and the number of entities in a table is very large.

Object-oriented database systems are better suited for applications containing more complex data with hierarchical substructures, a large variety of data types, and specialized operations and constraints on the data. Examples include computer-aided design systems, knowledge bases for expert systems, geographical modeling, robot control systems, and multimedia data analysis systems that combine audio, video, text, structured symbolic data, and numeric information.

The object-oriented database concept is relatively new, and currently available implementations do not support the full potential of the idea. More work is needed in implementing the full range of capabilities promised by the new approach, and for providing efficient realizations for the new capabilities. Theoretical studies indicate that object-oriented database management techniques should be more efficient than traditional ones in applications with large numbers of object types with relatively small numbers of instances per type. Traditional database implementation

techniques have been oriented towards small numbers of relations and large numbers of tuples per relation. New clustering and partitioning techniques are needed. Some steps in this direction are described in [Ref. 21] and [Ref. 22].

Object-oriented techniques hold great promise for more flexible applications, which can be built and modified with a small amount of effort. Libraries of reusable generic objects contribute to this goal. To alleviate the burdens of learning the contents of a large software base containing these reusable objects, expert systems for finding and combining available objects are being developed [Ref. 23].

In spite of the benefits discussed in this chapter, NMPC purchased the Model 204 RDBMS only several years ago, and has not made full use of that system yet. It is not possible to justify the cost of a new, object-oriented database management system. The new purchase would include such additional costs as personnel training, database systems support, and conversion costs, to mention only a few. The applications for which the DBMS will be used do not require the extensive features available in that type of system.



## **V. KNOWLEDGE BASED SOFTWARE ASSISTANT**

### **A. INTRODUCTION**

Advanced tools and programming environment capabilities supporting the software lifecycle are just becoming generally available. Many of these tools are focused on one aspect of the software development process. Additional capabilities that provide a high degree of coordination and communication among large numbers of software engineers, hardware engineers, and managers are required. The framework within which the software is developed plays a major role in realizing these capabilities. A Knowledge Based Software Assistant (KBSA) framework is being developed by Honeywell Systems and Research Center and is supported by RADC [Ref. 24]. This system provides an advanced framework containing a distributed knowledge base. The KBSA knowledge base can support data representation needs of tools, provide environmental support for the formalization and control of the software development process, and offer a highly interactive and consistent user interface. Programming environments for large-scale programming requires programming objects to be easily sharable between programmers, maintained with multiple versions, and protected by a user access policy. The use of a knowledge based programming system allows the machine to capture the software decisions

and the advanced reasoning mechanism may further assist with these decisions.

## **B. KBSA DESIGN**

The design objectives for the KBSA are: 1) support for the data representation needs of lifecycle tools, 2) support for coordination of activities that occurs during the software development process, 3) support for multiple levels of integration of tools, and 4) provide a highly interactive and consistent user interface.

The KBSA facet efforts are supported by three different frameworks. 1) **Socle** supports the Requirements Assistant and is a frames-based system that has been extended with mathematics constraints. 2) The **Common Lisp Framework (CLF)** supports the Specification Assistant and provides programming environment primitives used to represent programming knowledge. 3) **Refine** supports both the Project Management Assistant and the Performance Assistant and is an integrated language programming system. Each framework supports an object-oriented view of programming knowledge [Ref. 24].

The knowledge base manages the history of software development in the programming environment. Objects managed in the knowledge base are manipulated by the object schema, methods, rules, and demons. The object schema defines object classes containing a name, optional supertypes, and a list of slots. Methods define the messages accepted by the object class and

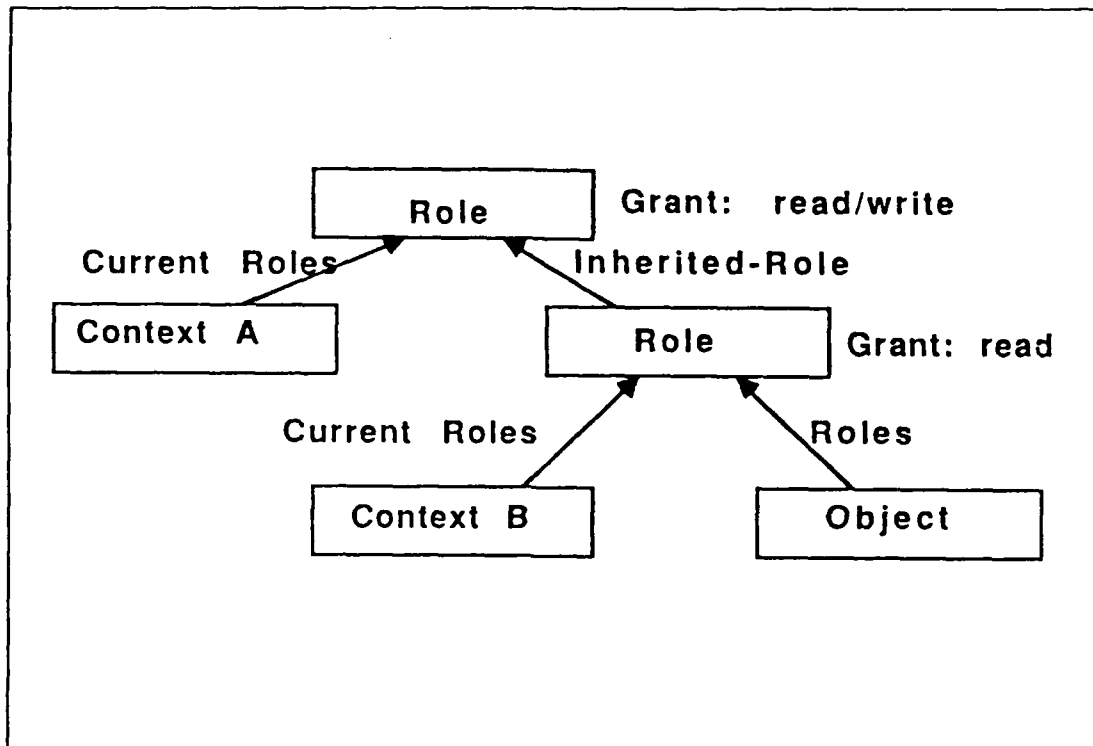
the operations that manipulate its instances. A demon associates an object method with a predicate condition and invokes a method whenever the predicate is satisfied.

The distributed object-oriented database is the basis for the KBSA. The programming environment support provided by the KBSA provides support in the following five areas: distributed knowledge base, access control, configuration control, transactions, and permanence.

The distributed knowledge base includes a wide range of information accessible and exchanged between users (programmers). The objects in the KBSA should be transparently managed and distributed by the system. Therefore, the user need not be concerned with the location of the data, duplication of the data, simultaneous operations on the data, and operation failures [Ref. 25]. Changes in the workspace by one programmer would cause immediate notification and provide real-time information about activities to all programmers using that workspace. The distributed objects are always manipulated by the framework.

Access control in any multiuser system provides protection for that system against unauthorized use. The framework in KBSA uses a scheme based on the capability model using two basic object classes: roles and contexts. A role is the relationship between a context and a knowledge base object. Each user is described with a single context object. If a context and an object both point to a given role, then the context has access

to the object using read or write permissions contained in the role. Access may be inherited along an inherited-role relationship.



**Figure 5** Access Control

In the example in Figure 5, Context A user is granted read/write access and Context B user is granted read only. The process has been designed to be flexible and efficient [Ref. 25].

Configuration control is the ability of the framework to manage multiple instantiations of a single object. Control of modifications and maintenance history of the objects must be maintained. Parallel development paths may be supported as described above in Chapter 4. A consistent configuration control policy is required.

Consistency of data must be maintained in a distributed system. Multiple operations on the knowledge base may cause an inconsistent view of the data until all operations have been applied. The system collects all operations into a single transaction to ensure the knowledge base does not attempt to resolve temporary inconsistencies until the transaction is complete. If there is an operation failure and a transaction is incomplete, the preceding operations are backed out until the whole transaction can be reapplied to the knowledge base.

Permanence is a consistent state of the knowledge base stored on non-volatile storage medium so that critical knowledge will not be lost in a system failure. Upon completion of a transaction, the knowledge base is brought up to date. In the case of a failure, the knowledge base can be brought back to the state it was in before the last transaction was started.

The user interface for the KBSA must be flexible enough to allow the beginner and expert programmers to navigate easily through the system and use the tools available. The user interface can be used in a graphical mouse-driven mode or a command interface mode.

The KBSA framework expands on existing technology in the following ways:

- 1) insures data integrity on behalf of tools it supports
- 2) provides associative access to data objects
- 3) provides distributed access to data objects on separate workstations
- 4) enables software activities to be formally described

- 5) provides for the management of changes to objects
- 6) provides a standardized set of user interface functions to tools

All of these capabilities could not be possible without the features of an object-oriented database.

The database can be viewed as a single object database accessible by all users on separate workstations. It synchronizes engineering operations that are performed against redundant stored copies of a data object. The knowledge base will support atomic transactions on the database to enable a set of operations to be handled as a single database state change [Ref. 25].

Several prototypes of the KBSA have been created to verify particular ideas with regard to a distributed object management system. The goal is to provide a programming environment with the additional capabilities mentioned above that also uses existing industry standards whenever possible. The prototype issues already explored are: logic mapping, constraint maintenance, access control, distribution, and permanence. The prototype issues which have yet to be investigated are: configuration management, transactions, and interfacing. Additional information on these prototypes can be found in [Ref. 25].

### **C. COMPARISON OF PROBE AND KBSA**

The KBSA and PROBE designs are quite different extensions of the common feature of an object-oriented database. This is one of the strengths of an object-oriented database system. The users are able to develop

extensions to the database which are specific to their needs. Both systems attempt to solve very difficult problems where solutions are not normally attempted in the database management area. The KBSA was designed specifically to support an integrated knowledge based programming environment, where PROBE attempts to remain a more general solution to the nontraditional problems in database processing. The PROBE system provides built-in support for geographical and spatial data. The KBSA frameworks provide support for software development applications in a distributed workstation environment.

#### **D. CONCLUSION**

There are a large number of programming tools on the market today. However, many of these tools do not provide any integration with other software development tools. High level frameworks can provide to the user the appearance of many tools to be closely integrated. In particular, the KBSA tools use many high level frameworks and consist of a large number of small operations. These operations are invoked by the state of the knowledge base rather than by the user. New integration techniques are required to join tools in the environment, integrating these tools is an effective means for providing users with the support for different programming needs and processes.

## **VI. TRANSITION AND INTEGRATION ISSUES**

### **A. INTRODUCTION**

In order to continue supporting the users that have become dependent on automatic data processing systems to perform their jobs, the systems on which they work must continue to perform adequately in their eyes. As soon as the systems are no longer supporting the users, they will no longer use them, but instead will find ways to work around them. As ADP equipment becomes more powerful, and software can perform more functions, the users of those applications will continue to use the systems more, and they will continue to think of more applications which need support through the software. Telecommunications between users will become more important as the daily functions of their jobs becomes more dependent on those lines of communication. It is necessary to continue to support the users of ADP systems with up-to-date hardware, software, and telecommunications. This chapter discusses issues for transition and integration through up-to-date or state of the art hardware, software, and telecommunications in order to continue support for the users of NMPDS.



## **B. HARDWARE**

Many Navy Commands are finally acquiring ADP support. However, quite often the equipment is outdated within a few years of acquisition. Navy ADP Managers are looking for ways to update their hardware without having to go through a long, drawn-out process. One way to upgrade Automatic Data Processing Equipment (ADPE) is to "piggyback" on a more general contract. For example, the Desktop II Air Force/Navy Zenith-248 microcomputers were purchased by various DoD departments through that type of contract. This section of the chapter will investigate conditions and definitions of a requirements contract, the procedures involved, and other facets of the agreement that NMPC must be aware. Alternative considerations are discussed, as well as advantages and disadvantages of "piggybacking" on a contract.

### **1. Background**

The ADPE acquisition process for large systems in the Federal Government can be a time consuming event. In 1979, the Distribution Support Division (NMPC-47) was tasked with developing a strategy to support the Distribution Department using automatic data processing equipment. At this time, the complete process was handled by passing volumes of paper from desk to desk. NMPC-47 developed a strategy which included two major efforts: ADPE acquisition and Software Development. Four AIS projects were conceived to manage officer distribution, enlisted distribution, training assignment reservations, and distribution management

support. The lengthy process involved to acquire ADP equipment required that NMPC lease hardware in order to develop the software. Milestone 0 approval for mission analysis and project initiation was gained, and contracts were awarded to two separate contractors to develop the projects. These contracts required the contractors to provide the leased hardware until the NMPC equipment was purchased. The Request for Proposal (RFP) was written and put on the streets in 1982 for a series of minicomputers to be located in Washington DC, New Orleans, LA, and Memphis, TN. The solicitation also included system software, database management, and hardware maintenance, among other things. Several bids were received, and evaluation of the offers commenced. A Source Selection Evaluation Board (SSEB) was formed using six members of NMPC-47 and tasked with evaluating the bids. This process was estimated to be a one year process, and turned into a three year process. Each step of the way was held up for one reason or another. The RFP needed changes as NMPC gained a better understanding of the projects. The six members of the SSEB were still required to perform their regular jobs in addition to the duties of the SSEB. Therefore, they worked on each job half of the time. The value of the contract was \$28 million, and so the bids required very careful evaluation. No mistakes could afford to be made, and this slowed the process. Contract award was initially planned for April 1983, and it was finally awarded in April 1985.

In the meantime one project had begun development on a leased Prime, been moved to a leased IBM, and finally was migrated to the permanent NMPC IBM hardware. Another project had been developed on a IBM 3033 run by the Navy (NMPC-16). Disk storage and terminal access for both programmers and users proved markedly constrained by the production priorities on the NMPC-16 equipment. Software development for this project was much slower than planned. The third and fourth projects were developed on a leased Vax 11/780. One of the subsystems of the project went into production while on the leased Vax and still remains there. Priorities set by NMPC-47 have not allowed for the time to convert the software to run on the Navy owned IBM machines [Ref. 26].

The software used to develop these projects, Sage's Advanced Programming System (APS) is a COBOL code generator, and enables the programmer to produce COBOL code very quickly. APS was originally planned to be fully portable from one machine to another. However, with upgrades in the versions of APS, the company eventually decided to support only the IBM compatible version of APS. Therefore, the software developed on the Vax remained in the older version of code.

The amount of money spent on leased hardware and on converting software to newer versions of APS is in excess of \$3 million. The excessive amount of money spent, compatibility problems between the projects, migration problems moving to different hardware, and coordination problems among the many players were primarily caused by the delays in

the hardware acquisition process. Even once the contract was awarded in 1985, only partial funding for the equipment was provided. Funding shortfalls prevented delivery of all of the hardware awarded in the contract. The original strategy planned for hardware upgrades and project redesign to begin in 1990. If a similar acquisition strategy is used for this upgrade, the new hardware may not become a reality until the late 1990's. Therefore, NMPC-47 must use a new approach in order to continue support for their users. By using a contract vehicle already in place, NMPC can concentrate the efforts on receiving the funding approvals necessary to purchase the new hardware.

## **2. Requirements Contracts**

A requirements contract is an express contract where an agreement with all the detailed terms and conditions are clearly stated. The contract specifically lists all requirements which must be met by the contractor. These requirements are listed in the "specifications" of the contract, and are mandatory terms required by the offeror.

The Desktop II contract awarded to Zenith on February 28, 1986 by the Air Force is a firm, fixed price, requirements contract. The contract with Zenith Data Systems provides for the purchase of Z-248 desk top microcomputers.

The basic Z-248 system on the contract has an 80286, 8MHZ central processing unit with 512KB random access memory, dual 360KB floppy disk drives, various ports, enhanced display adapter, MS-DOS 3.1 and MS-WINDOWS. The memory can be upgraded to 1.1MB and then to 3.1MB, and up to two internal 20MB hard disks can be added to the system. Other peripherals available on

the contract are dot matrix/letter quality printer with cut sheet feeder, color graphics printer, graphics plotter, graphics input device, RGB color monitor, monochrome monitor, dial-up 2400 baud modem, 20MB tape backup system, and an 80287 8MHZ arithmetic coprocessor. A selection of system software, application software, computer aided instruction packages, and various emulators is also available on the contract. [Ref. 27]

This contract was an all encompassing contract which provided hardware, software, and maintenance for the purchasing of microcomputers. It provided a single contract vehicle for the agencies to use in order to acquire all needed services.

Activities in the Navy, Air Force, and Defense Logistics Departments are required to use this contract when general purpose, stand alone workstation applications can be met by this contract. The contract is not mandatory, but may be used by the Army, OSD, and other Department of Defense Activities [Ref. 28]. The term of the contract is for 12 months, with two additional 12 month options. Both 12 month options were executed, and the contract expires on Feb. 28, 1989.

The Desktop II contract was very successful in the eyes of both the Department of Defense and Zenith Data Systems. According to Jerry K. Pearlman, president of Zenith Electronics Corporation, the success of the contract was "by providing the government with the best life cycle costs." Zenith was required to develop unique hardware for the government, and they used it as a platform to grow into other channels of distribution. The technical features developed for the government have become key building blocks for their commercial strategy. For example the government pushed

for a lighter and smaller laptop, and Zenith scrambled to develop a very small package. This package became successful commercially as the SupersPort 286 package [Ref. 29].

The success of the contract was not anticipated by DoD officials. The Government Services Administration (GSA) granted procurement authority for 90,000 microcomputers for the 3 year life cycle of the contract. As of January, 1988, defense personnel ordered more than 225,000 Z-248 microcomputers [Ref. 30]. The advantage was that the contract was already in place, and agencies were able to get their microcomputers quickly and efficiently through the contract. Microcomputers are playing a bigger role in the DoD by performing functions that used to be done on minicomputers and mainframes.

### **3. Procedures**

A primary advantage of using a requirements contract already in place is the ease of purchasing the equipment. Procedures set forth in the Desktop II Zenith contract were relatively simple to follow. This feature in itself made the acquisition of microcomputers through the Desktop II contract very attractive to the various agencies. Points of Contact were assigned in each agency to establish procedures and to help users with the equipment purchases.

The Naval Data Automation Command (NAVDAC) designated the Naval Regional Data Automation Command (NARDAC), Norfolk, as the single POC for consolidating delivery orders from Navy activities to Zenith

Data Systems. The procedures for the Navy were outlined in a NAVDAC Advisory Bulletin as follows:

- 1) Forward a requisition to the local ordering office for the creation of a delivery order (DD Form 1155) citing the required Contract Line Item Numbers (CLINs) and prices.
- 2) Ensure a POC name and telephone number at the ordering office are included.
- 3) Include a technical user POC name and telephone number for receipt of any advance information regarding technical and administrative handling of equipment upon delivery.
- 4) Partial deliveries are allowed only at discretion of the Government. If the user desires, the delivery order must state which items will be an allowable partial delivery.
- 5) Include necessary shipping instructions.
- 6) The order should indicate security clearance required for on-call on-site maintenance personnel.
- 7) A maintenance plan specified in the contract must be designated on the delivery order.

The ordering office forwards the DD 1155 to NARDAC Norfolk along with a Comptroller of the Navy Form 2275 for 2 percent of the total delivery order. NARDAC consolidates the orders, logs them, and forwards them to Zenith Data Systems. Microcomputer technical assistance is provided by each of the NARDACs. This assistance includes configuration support, training classes, applications development, networking, and interconnection services [Ref. 27]. Many activities purchase hardware without receiving the proper training and support. This service offered by the NARDACs is very valuable. In addition to the support mentioned above, NARDAC Norfolk

produces a monthly magazine, CHIPS, specifically designed to help the Navy microcomputer users gain a better understanding of their machines.

#### **4. Desktop III**

The Desktop III contract is the follow-on contract to the Desktop II Zenith-248 contract. It includes specifications for a high-end, next generation microcomputer. These systems will be 16 or 32 bit processors built around either the Intel 80386 chip or the Motorola 68000 series chip technology. The machine should be compatible with the enormous amount of Z-248s currently owned by the Department of Defense. The microcomputers may be MS-DOS based, but also will provide for the UNIX and OS/2 operating systems [Ref. 31]. Some other features discussed are removable hard disk technology, shipboard power spike protection, and requirements for modems to be used outside the United States.

A recent article in Government Computer News announced another extension in the deadline for proposals for Desktop III. Original deadlines were set for November 1988 and original contract award was expected to be made in February 1989. The Air Force Standard System Center (SSC) announced the extended deadlines to be March 3 and March 13 for the two separate parts of the proposal. An award date has not yet been determined. Once proposals have been received, an estimated time for evaluating the bids will be given. The last announcement by the Air Force projected August 1989 as an estimated award date. No changes were made to that date in the recent announcement. The SSC announced the seventh



amendment to the RFP on January 26. The extension was given to allow vendors time to incorporate changes in their proposals due to the amendments [Ref. 32].

A memorandum of agreement was signed several years ago by the Air Force, Navy, DLA, and other Defense agencies and cooperation among these services will continue on both the program and project levels. Management representatives and contract officers from the agencies meet on a quarterly basis at the program level. Project managers brief the program managers on their specific projects, including Tempest computers, Z-248s, multiuser systems and laptop computers [Ref. 31]. The Air Force Small Computer Center, a division of SSC, is the program office for Desktop III and communicates with its user community by sending out surveys to determine what people will need in their desktop computers. When the answers are received, the center consolidates them and sends the responses back to the users for review and comment. Preliminary specifications are written and sent out for another look. With a user community as large as the Department of Defense, the Air Force is faced with a difficult task of including all user requirements. This has contributed to the large number of modifications to the Desktop III RFP.

The procedures for ordering hardware and software will be similar to the Desktop II procedures. These procedures have been working smoothly for the past four or five years, and so there are no anticipated changes.

## 5. Alternative Considerations

Once minicomputer machines have reached maximum utilization, steps must be taken in order to continue servicing the user community. One option that will provide support is to purchase larger or more powerful machines. These may be either mainframe or minicomputers. The cost of purchasing these type of machines usually exceeds the \$250K for noncompetitive buys and the \$2.5 million for competitive buys authorized by GSA under the blanket delegation of procurement. Therefore, competitive procedures according to all instructions must be followed. The process may take several years, as mentioned in the background section of this chapter. Many situations can not withstand this excessive time limit, and mainframe and minicomputer solutions must be examined carefully. Powerful microprocessors are allowing users to perform more and more functions on smaller and less expensive machines. Unless the applications require the processing power of a large machine, a mainframe may not be necessary.

A readily available source of purchasing microcomputers is via the GSA schedule. The GSA negotiates contracts with vendors in local areas to provide federal agencies with products at a reasonable price. The items listed on the GSA schedule contracts are usually discounted to a price lower than that offered to the commercial customer. However, these items may or may not meet the special needs of the military user. Requirements contracts are able to specify these unique requirements for the users, and

are still able to negotiate prices lower than the commercial market. The GSA schedule contracts provide a reasonable means of acquiring microcomputers if a requirements contract is not currently in place.

A viable solution that can provide more processing power to the user is to upgrade the microcomputers already in place. Motherboards and additional memory boards are readily available on the market today. These options can upgrade machines with the 8088 microprocessor, 8086 microprocessor, and the 80286 microprocessor. For the older machines, the upgrades required to gain the speed provided by machines built around the 80386 microprocessor may be extensive, i.e. screen/graphic displays, printers, modems, etc.. In some cases, it may be more effective to purchase a complete new package as a whole unit in order to provide the user with equipment that will still be effective in the near future. These options must be considered carefully. The growing number of microcomputers in the Defense Department and the shortages in money will force managers to find innovative ways to improve the life cycle of their microcomputers.

## **6. Conclusions**

Working together has provided the Department of Defense several benefits through their memorandum of agreement made a few years ago. By combining common functions and having unique requirements placed on them, the agencies can negotiate contracts to best meet their needs.

One specific area of benefit to the agencies is the cost savings. Only one organization is tied up with writing specifications, soliciting

proposals, evaluating bids, and administering the contract. The other agencies are not required to spend time or other valuable resources in duplicative efforts on negotiating a contract. This has probably saved the Department of Defense millions of dollars in man-hours alone.

Cost savings can also be realized by using a requirements contract for workstation applications to purchase microcomputers that provide the additional processing power needed. The typical mainframe or minicomputer solution to the processing power problem can be much more costly. The needs of the users can be matched more closely to the amount of hardware purchased.

Another benefit to using a requirements contract that is already in place is the quick delivery. Without the added time delay negotiating the contract, all efforts may be placed on obtaining the funding required to purchase the equipment. Once funding is approved and allocated, the delivery order may be typed immediately. Once NARDAC receives the order, in this particular case Zenith will deliver the ADPE within 30 days: A marked improvement over four years. An additional benefit in this area is the consistency of personnel involved in the decision making process, especially the "upper management". The projects sponsored by NMPC were strongly supported by high-level management in the early stages. However, as time progressed, the personnel also moved on to new positions. Each new flag officer required a briefing of the projects. Priorities often changed, and so funding may or may not have been sought after as

fervently. Therefore, the sooner hardware can be purchased, the more likely project offices can avoid changes in mid-course.

A final benefit to using a requirements contract is the "hassle free" environment. The individual agencies and organizations are not administering the contract. There is no need to be concerned with the legal matters of the contract administration. The procedures for purchasing require each agency to use one point of contact in dealing with the vendor. This organization is the only one that needs to be tied up with the minute details. Equipment maintenance has already been negotiated, and the agencies must adhere to the agreements set up in the contract. Most of the bureaucratic red tape has been taken off the shoulders of the individual activities. Contract "piggybacking" seems to be an excellent way for organizations to move to the next generation.

### C. SOFTWARE

The strategy that must be applied in order to keep NMPC running in an efficient software environment is a bit more vague than the strategy of the hardware environment. Unfortunately, this is one of the downfalls of the software development process in general. Guidelines can be set up to be followed, but in general the process is vague and subject to the definition of the players involved. Unfortunately, in the case of a Navy command, the players are continually moving, and so the definitions of the elements in the software development keep changing. Therefore, the best strategy for

developing or upgrading software is to do so in a modular fashion. Distributed database management systems are showing many benefits over centralized database systems as described in Chapter 3. By developing the systems a small chunk at a time, only a small number of players need be involved. This may reduce the amount of midstream changes in the process.

The reorganization of NMPDS will provide an excellent opportunity for redesigning the software. For instance, the Assignment Management Support System (AMSS) may be an excellent starting point. Convert one of the smaller modules of AMSS from using flat files into a module using a DBMS. The applications will change as the functions available from the DBMS provide more and better capabilities. Functions that are now being performed differently in the officer and enlisted systems may be combined into single functions. As the system modules are redesigned and the hardware is purchased, the modules may be downloaded to the microcomputers or workstations and run in a distributed environment.

The design of the database files will be a primary issue in the software redesign effort. The Data Administrator, Database Administrator, System Administrator, and Project Managers should work very closely together in order to design the most efficient database system. In the early development stages of NMPDS, there were no statistics available from the hardware systems administration, since each project was developed on different leased machines. Now that there are some statistics available from

the current NMPDS hardware, these statistics may provide a guide for the design of the database files. For example, statistics have shown that there is a high rate of input/output operations which severely slow the current system. By designing the database files with these statistics in mind, a management process may be designed to handle the input/output operations more efficiently.

Database management system performance should be an issue considered meticulously along each step of the redesign effort. If performance issues are not addressed initially, they will quickly become problems once the system is in full use by the application users. Computer Corporation of America's (CCA's) objective was to design Model 204 Database Management System to be an "advanced data model for rapid retrieval of information from very large databases." [Ref. 33] The Model 204 DBMS design provides flexible and efficient data base manipulation. The number of tests run by CCA have shown excellent response times for a large number of complex queries against extremely large databases. "Model 204 has accommodated as many as 800 simultaneous users accessing a 15 gigabyte database with less than two second average response time." [Ref. 33] By using a DBMS designed for efficient processing in a distributed environment, like Model 204, then performance problems will be delayed in comparison to using a more traditional solution.

The NMPDS system maintains a large number of interfaces with external systems. The four current systems use input from at least six

different organizations, and each project produces output to a number of other activities. These external activities may or may not be currently running under database management systems. The interfaces between the different systems may develop into a major issue. The Chief of Naval Operations (OP-162) sponsored a study on seven Manpower, Personnel, and Training (MPT) or research-based AISs, the database management systems used by those AISs, and the level of effort anticipated to communicate data between the various DBMSs.

The draft response written by Oak Ridge National Laboratories (ORNL) discusses the interface issues between the relational model DBMS (RDBMS) and the non-relational database systems in the group. Of the RDBMSs, some of the languages used are SQL-based, and some are not. This provides an additional interface issue between the systems. The ORNL response suggests an Interface Processor (IP) to provide communications between the query languages and the query responses from any one of the seven DBMSs to any other. Each organization has a mono-model, mono-lingual DBMS, also referred to as a homogeneous DBMS [Ref. 34]. Current research efforts in the "heterogeneous DBMS" field are showing excellent results for future DBMS communications. Each of the seven DBMS systems mentioned by ORNL are homogeneous DBMS which do not provide database access across models or automatic translation of the query languages. In [Ref. 34], both software and hardware solutions are presented for heterogenous DBMS environments. The software solutions



provide better overall data sharing, control, and utilizations through data model transformation, data language translation, and cross model accessing. The hardware solutions suggested in the paper provide lower maintenance costs and better support of upgrade capabilities. Some variations of these types of solutions are on the market, and as the need for inter-DBMS communications increases, so will the availability of commercial products.

An additional interface issue for NMPDS is with the Military Personnel Record Data Management Department (NMPC-3). A project initiated by NMPC-3, called System 90, has been proposed to convert the enormous amount of military personnel microfiche records to digital data on optical disk storage. The microfiche contain exact replicas of the personnel records, and are used for official archival purposes. Some of the documents contained in the records have signatures, which cannot officially be reproduced through text representation. Therefore, "pictures" of the personnel records must be recorded, either through microfiche or optical disk storage, in order to meet U.S. legal requirements. The microfiche contain fitness reports which are used by the Distribution Department (NMPC-4) along with other information from the microfiche in order to manage naval manpower resources and fill world-wide vacant Navy billets. A pilot system has been proposed which will meet both the needs of NMPC-4 and that test essential characteristics of the proposed system in its entirety. The System 90 pilot system will use optical/laser disk storage which will allow the storage and retrieval of all personnel records

mentioned above. It will require on-line access for terminal, personal computers, or workstations with very high resolution screens to allow display of the optical images. The hardware and software development plans for both organizations should include and combine the requirements specifications as well as the funding requirements of each organization. The hardware proposed in the first section of this chapter meets both organizations' needs.

#### **D. TELECOMMUNICATIONS**

The current NMPDS telecommunications configuration can be divided into two categories: local area and wide area. The local area network installed in Washington, DC, is a broadband coaxial cable. Long distance communications include 56 KB lines and 9.6 KB lines leased through public data services. The local area network was initially installed as a "Beta Test site" for OPNAV (OP-16) in 1984 and 1985. With statistical multiplexing and full duplex features, it can provide service for up to 1500 users. Telecommunications for long distance applications will be via the Defense Data Network as soon as that system is installed at all NMPDS sites.

The future needs of NMPDS in terms of telecommunications are to be able to: 1) download or upload database files from the minicomputers to workstations, 2) provide real-time distributed data between the local users where required, 3) provide synchronized data among long distance sites, and

4) provide contingency operations in case of failure of a site or telecommunication line.

The minicomputers resident in Washington should act as network servers, providing communication protocols, broadcast strategies, and overall coordination of the network. Some possible distributed solutions were discussed in Chapter 3 and can be applied by using characteristics of the desired system to choose an appropriate network strategy. There are only a certain number of data elements in NMPDS that require real-time update distribution among local users.

The Officer Assignment Information System, with the personnel, activity, billet, and policy files, for the most part does not require that changes to data be distributed immediately. A person in the personnel file can be assigned by only one detailer, a billet can be approved by only one placement officer, and so for the general daily operations, real-time distribution of updates is not required. Reservations to seats in training classes, on the other hand, can be executed by several detailers. Here is a requirement to broadcast the changes of the data elements with respect to the seats available in a particular class, or to send all queries to a central location for each particular class. A network strategy must be designed to alert local users when seats have been reserved, and to update each users' local database. At the time of database design, the elements that require real-time update should be identified.

The full duplex feature of the local area network allows simultaneous two-way communication signals on the cable. This feature is necessary to provide real-time update to the more than 500 local users of NMPDS. At least for the next five years, the local area network has sufficient capacity to support the developments of NMPDS.

## E. CONCLUSION

The primary transition and integration issues for NMPDS to move to a distributed database management system will be in terms of hardware, software, and telecommunications. The best choice for acquiring the necessary hardware is to use a requirements contract such as the Desktop III contract to purchase high end microcomputers that may be used for workstation applications. The software redesign efforts should be done in a modular fashion while concentrating on issues such as database performance and integration with external database systems. Every effort should be made to combine the NMPDS requirements with other organizations like NMPC-3 to gain the maximum utilization of new hardware and software technologies. The telecommunications efforts combine some of the hardware and software issues of NMPDS. However, emphasis in this area should be placed on designing an appropriate network communication protocol to meet the specific needs of NMPDS.

## VII. CONCLUSION

### A. SUMMARY

This thesis investigates the implementation of a distributed object-oriented database management system. By using the Naval Military Personnel Command (NMPC-47) automated information systems as a case study, the research explored the possibilities, requirements, and issues related to distributed database management systems (DDBMS) and object-oriented database management systems (ODBMS). A variety of distributed methods, algorithms, and protocols are presented and discussed. The features and benefits available from work in the field of object-orientation are discovered and reviewed. The Knowledge Base Software Assistant (KBSA) is a distributed system developed by Honeywell Systems and Research Center for Rome Air Development Center. This project explores the concepts of using a knowledge base framework to provide a complete and integrated programming environment for software development of large systems. By using a distributed object-oriented database management system, the project goals have been met. Transition and interface issues are then discussed for NMPC to consider in terms of hardware, software, and telecommunications issues.

## **B. RECOMMENDATIONS**

A distributed environment has been demonstrated by many research reports as the most efficient and practical way to process information in the future. Systems may be distributed in terms of where the data is located, or in terms of where processes are executed. An appropriate algorithm must be chosen that specifically meets the needs of the system being distributed.

The features that an object-oriented DBMS can provide include extensibility, software reusability, and representation of complex data structures. Applications developed using an ODBMS are revealing even more benefits than originally expected. However, this concept is still relatively new, and currently available implementations do not support the full potential of the idea. Until more research can be completed in this area, NMPC should not consider changing from the DBMS currently in use by that organization.

The most appropriate way to implement the distributed environment at NMPC is through the use of microcomputers as workstations. The hardware can easily be obtained through joint service requirements contracts. Distributing the data and processes onto each users' workstations alleviates the tremendous workload currently on the NMPC hardware.

NMPC has obtained \$1.1 million FY90 OPN funds for hardware upgrades, [Ref. 35]. A portion of that OPN funding, \$15-20k should be earmarked to purchase five machines through the Desktop III contract. The estimated price of the microcomputers is \$3-5k. Desktop III contract award

is anticipated for early FY90, and the processing of the hardware acquisition can occur immediately. These five machines should be distributed within NMPC-47 as follows: one machine to the N-47 technical advisor, one machine to the N-47 database administrator, and three machines to the software development team. These microcomputers should not be obligated to specific people, but rather should be made available whenever development of this project is concerned. These machines would initially be used for testing the new software, testing download and upload processes, testing machine capabilities, and evaluating system specifications. By using these machines early in the development process to gather system performance statistics, more accurate recommendations can be made by the technical advisor for the future of NMPDS.

The software development process should begin first with conceptual design of the system. Approximately one month should be used to decide how the new projects should be organized. For example, AMSS will consist of the functional modules currently in OAIS, EAIS, and SPIRIT. There are functions in each project that are unique to the application, and there are many functions that are the same, or similar across the AISs. Each project requires information from personnel records, but the officers and enlisted look at different information. An Officer Personnel Information Module (OPIM) and an Enlisted Personnel Information Module (EPIM) may be the first two modules converted to the Model 204 DBMS. Additional modules may include both officer and enlisted versions of : 1)

an Assignment Decision Module (OADM and EADM), 2) a Training Track (OTT and ETT), 3) and Order Production Module, and 4) other modules appropriate for this system.

Once conceptual design for the first project is completed, database design must begin. The database design process should include the following players: the Data Administrator, the Database Administrator, the Project Manager, the Technical Advisor, and 2 or 3 programmers. If the team is much larger than this, the productivity within the group will begin to go down. For the first few months, it may be advisable for this group to meet away from their regular workspace to avoid interruptions from their jobs.

The order for which the software should be converted may follow along with the original development plans of the projects. For example, the Surface Warfare officer detailers were the first users involved in OAIS. By dividing the modules into applications that work for smaller groups of users, the process is more successful.

The initial module should take approximately six months to be developed, and then the module should be thoroughly tested. The major problems with the software will be identified and resolved. At this time, more programmers should be added to the development process. The programmers should be divided into teams of 3 or 4 programmers to work on these modules. The teams should consist of a mix of navy programmers from the NMPC-47 Information Center and contractor personnel. Modules



are designed by the original team mentioned and assigned to programming groups for development.

This conversion will require a major effort. Although the level of effort will not be equal to that of automating a manual process, the support required to accomplish the conversion will be a necessity. Funding to support the software conversion must be approved. Management must assign this process a top priority in order to gain the support and enthusiasm from all personnel. Cost benefits analysis should be completed to back up the benefits proposed by the conversion. Any new development efforts should only be done in Model 204 after the beginning of FY90.

Throughout the software conversion development process, the users should be intricately involved. User group meetings have been used in the

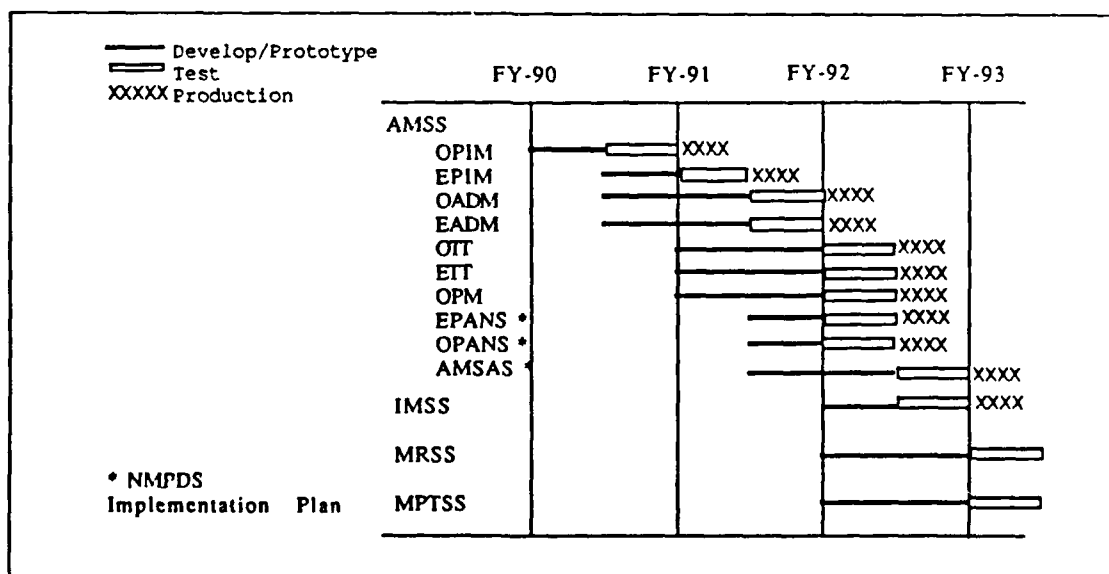


Figure 6 NMPDS Conversion Plan of Action

past by NMPC-47 to communicate with the users during software development. These user groups now consist of representatives of all users. This large group may encumber the development process. Trying to incorporate all user requirements from the beginning can often bog down the development process to the point that no software may ever be placed into production. With this in mind, a few officer detailers from the Surface Warfare Division should represent the user community upon initial development. Figure 6 is a chart of the estimated timeline for the conversion of NMPDS to a distributed DBMS.

## LIST OF REFERENCES

1. Distribution Support Division (NMPC-47), Naval Military Personnel Distribution System Implementation Plan, Department of the Navy, May 31, 1988.
2. Oak Ridge National Laboratory, "Draft-Evaluation of Need for Converting NMPDS Data Files to a DBMS," Unpublished paper, 1988.
3. McGovern, J., "DCA-A Distributed Communications Architecture", Evolutions in Computer Communications, pp. 359-366, North-Holland, New York, 1978.
4. Saito, M., Kawazu, S., Izuoki, T., and S. Ishigaki, "Concept and Design Considerations of Hierarchical Database Access Protocol for Distributed Database", Evolutions in Computer Communications, pp. 817-821, North-Holland, New York, 1978.
5. Snyders, Jan, "Definition of 'Distributed' Depends on Whom You Ask," Government Computer News, pp. 72-74, August 29, 1988.
6. Hayashi, T. and Sakai, T., "Considerations on the Design of a Network-Oriented Operating System", Evolutions in Computer Communications, pp. 239-244, North-Holland, New York, 1978.
7. Ono, K., Urano, Y., Suzuki, K., and Kurematsu, A., "Distributed Communication Processing in a Packet Switched Computer Communications Network", Evolutions in Computer Communications, pp. 657-662, North-Holland, New York, 1978.
8. Yatsuboshi, R. and Tsuda, T., "An In-House Network Configuration for Distributed Intelligence", Evolutions in Computer Communications, pp. 155-160, North-Holland, New York, 1978.
9. Booth, G.M., "Honeywell's Distributed Systems Environment", Evolutions in Computer Communications, pp. 347-351, North-Holland, New York, 1978.
10. Computer Corporation of America, Technical Report CCA-88-01, OVERVIEW OF SHARD: A SYSTEM FOR HIGHLY AVAILABLE REPLICATED DATA, by Sarin, S., DeWitt, M., and R. Rosenberg, Cambridge, May, 1988.

11. Ladkin, P., Specification of Time Dependencies and Synthesis of Concurrent Processes, Kestrel Institute, 1987.
12. Zhae, Wei and Ramamritham, K., Distributed Scheduling Using Bidding and Focused Addressing, The University of Massachusetts, Amherst, 1985.
13. Chu, Wesley W. and Leung, K., Task Response Time Model & Its Applications for Real-Time Distributed Processing Systems, The University of California. Los Angeles, 1984.
14. Apers, Peter, "Data Allocation in Distributed Database Systems," ACM Transactions on Database Systems, Vol. 13, No. 3, pp. 263-304, September, 1988.
15. Musgrave, Bill, "Network Management: Keeping the Connection", Datamation, vol. 33, no. 17, pp. 98-107, September 1987.
16. Berry, DPCM(SW) R., of the Naval Military Personnel Command, Electronic letter to the author, February 21, 1989.
17. Cox, Brad J., Object-Oriented Programming - An Evolutionary Approach, Addison-Wesley, Reading, MA, 1986.
18. Orenstein, J., Goldhirsch, D., and Manola, F., "The Architecture of the PROBE Database System", PROBE Project Working Paper, Cambridge.
19. F. Manola, and Orenstein, J., "Toward a General Spatial Data Model for an Object-Oriented DBMS", Proceedings Twelfth International Conference on Very Large Databases, Kyoto, Japan, August 1986.
20. Berzins, V., Robust Generation of Unique Identifiers in Distributed Systems, Naval Postgraduate School, Monterey, CA, June 1987.
21. Berzins, V., "Cache Management in Software Engineering Databases", Proceedings on the First International Workshop on Computer Aided Software Engineering, Cambridge, pp. 523-528, May 1987.
22. Ketabachi, M. and Berzins, V., "Mathematical Model of Composite Objects and its Application for Organizing Efficient Engineering Data Bases", IEEE Transactions on Software Engineering, January 1988.
23. Luqi, "Knowledge Base Support for Rapid Prototyping", IEEE Expert, vol. 3, no. 4, pp. 9-18, November 1988.

24. Huseth, S., and King, T., A Common Framework for Knowledge-Based Programming, Minneapolis, April, 1988.
25. Huseth, S., and others, KBSA Framework Final Technical Report Phase 1, Minneapolis, July 18, 1988.
26. Wyrick, Lynn, System Decision Paper for the Support Programs for Incentives, Retention and Training Assignments, p. 4, May 1987.
27. Naval Data Automation Command, NAVDAC ADVISORY BULLETIN, Bulletin No. 77, Washington, DC: Department of the Navy, May 15, 1986.
28. Air Force Computer Acquisition Center, AFCAC Contract No. F19630-86-D-0002, February 28, 1986.
29. Kirchner, Jake, "Government Leads the Way, Zenith Chief Says," Government Computer News, p. 10, December 19, 1988.
30. Bass, Brad, "AF Suspends Z-248 Buys," Government Computer News, pp. 1-4, January 8, 1988.
31. Danca, Richard, "After Z-248, What Next?", Government Computer News, p. 8+, April 15, 1988.
32. Danca, Richard, "Air Force Moves Desktop III Proposal Deadline to March," Government Computer News, p. 8, February 6, 1989.
33. Computer Corporation of America, Model 204 System Overview, Cambridge, 1987.
34. Hsaio, D., and Kamel, M., Heterogeneous Databases: Proliferations, Issues and Solutions, Naval Postgraduate School, Monterey, CA, January 1989.
35. Stoopman, G., of the Naval Military Personnel Command, Electronic Letter to the author, March 1, 1989.

## BIBLIOGRAPHY

Fukuda, Z., "A Study of Centralized and Distributed Processing", Evolutions in Computer Communications, North-Holland, New York, 1978.

Ketabachi, M. and Berzins, V., "Modeling and Managing CAD Databases", IEEE Computer, vol. 20, no. 2, February 1987.

Stallings, W., Data and Computer Communications, 2nd ed., Macmillan, New York, 1988.

Wei, S. and G. Lee, "Extra Group Network", Computer Architecture News, Washington, DC, 1988.

Wernikoff, S., "An Overview of Database Access and Remote Computing Communications Service", Evolutions in Computer Communications, North-Holland, New York, 1978.

## INITIAL DISTRIBUTION LIST

- |    |   |    |
|----|---|----|
| 1. | Defense Technical Information Center<br>Cameron Station<br>Alexandria, VA 22304-6145  | 2  |
| 2. | Library, Code 0142<br>Naval Postgraduate School<br>Monterey, CA 93943-5002  | 2  |
| 3. | Office of Naval Research<br>Attn: CDR Michael Gehl (Code 1224)<br>800 N. Quincy Street<br>Arlington, VA 22217-5000                              | 1  |
| 4. | Naval Military Personnel Command<br>Attn: Mr Gerry Stoopman (NMPC-47)<br>Washington, DC 20370   | 10 |
| 5. | SYSCON Corporation<br>Attn: Mr. Robert Kidwell<br>1000 Thomas Jefferson, NW<br>Washington, DC 20007   | 1  |
| 6. | SYSCON Corporation<br>Attn: Mr. Robert Lovell<br>1000 Thomas Jefferson, NW<br>Washington, DC 20007  | 1  |
| 7. | Xerox Advanced Information Technology<br>Attn: Ms. Rita Bergman<br>King Street Station<br>1800 Diagonal Road, Suite 300<br>Alexandria, VA 22314 | 1  |
| 8. | Dr. Valdis Berzins<br>Code 52BE<br>Naval Postgraduate School<br>Monterey, CA. 93943-5004  | 2  |
| 9. | Fleet Combat Directional Systems Support Activity<br>Attn: Lt. Lynn Wyrick<br>San Diego, CA 92147-5081  | 2  |

10. Honeywell Systems and Research Center 1  
Attn: Mr. Steve Huseth  
3660 Technology Drive  
Minneapolis, MN 55418
11. Naval Military Personnel Command 1  
Attn: Mr. Mike McNeill (NMPC-16F)  
Washington, DC 20370
12. Chief of Naval Operations 1  
Attn: Dr. Earl Chavis (OP-162)  
Washington, DC 20350
13. Mr. George Dailey 1  
Oak Ridge National Laboratories  
Oak Ridge, TN 37830
14. Naval Regional Data Automation Command 1  
NARDAC, Washington  
Attn: Mr. Dennis Hardy (Code 42)  
Washington, DC 20374
15. Enlisted Personnel Management Center 1  
Attn: Mr. Lew Cornett (Code 30)  
New Orleans, LA 70159
16. Naval Military Personnel Command 1  
Attn: Capt. R. J. Hayes (NMPC-4B)  
Washington, DC 20370
17. Naval Military Personnel Command 1  
Attn: Capt. T. I. Eubanks (NMPC-00B)  
Washington, DC 20370
18. Chief of Naval Operations 1  
Attn: Dr. R. M. Carroll (OP-01B2)  
Washington, DC 20350
19. Naval Research Laboratory 1  
Attn: Dr. Elizabeth Wald (Code 5150)  
Washington, DC 20375-5000
20. Navy Ocean System Center 1  
Attn: Linwood Sutton (Code 423)  
San Diego, CA 92152



- |     |   |   |
|-----|---|---|
| 21. | National Science Foundation<br>Attn: Dr. William Wulf<br>Washington, DC 20550   | 1 |
| 22. | Defense Advanced Research Projects Agency (DARPA)<br>Integrated Strategic Technology Office (ISTO)<br>Attn: Dr. Jacob Schwartz<br>1400 Wilson Boulevard<br>Arlington, VA 22209-2308 | 1 |
| 23. | Defense Advanced Research Projects Agency (DARPA)<br>Director, Naval Technology Office<br>1400 Wilson Boulevard<br>Arlington, VA 22209-2308   | 1 |
| 24. | COL C. Cox, USAF<br>JCS (J-8)<br>Nuclear Force Analysis Division<br>Washington, DC 20318-8000   | 1 |
| 25. | U.S. Air Force Systems Command<br>Rome Air Development Center (RADC)<br>Attn: Mr. Samuel A. DiNitto, Jr.<br>Griffis Air Force Base, NY 13441-5700                                   | 1 |
| 26. | U.S. Air Force Systems Command<br>Rome Air Development Center (RADC)<br>Attn: Mr. William E. Rzepka<br>Griffis Air Force Base, NY 13441-5700  | 1 |